This section includes articles and white papers from a variety of sources both internal and external to Microsoft. Categories include:

® Technical Articles

® Technical White Papers

This section includes a list of books available from Microsoft Press and sample chapters from a few Microsoft Press books.

{ewc MVIMG, MVIMAGE,!welcome.shg}

Return to Welcome Screen

This CD-ROM-based training course teaches you how to create Web applications with Microsoft Visual InterDev.

Course Content

The content for this course is organized into the following ten chapters.

Chapter 1: Planning a Web Site

This chapter introduces you to the technology used in the course. You will learn how each member of a Web development team contributes to the process of planning and creating a Web site.

Chapter 2: Developing a Web Project

In this chapter, you will learn how to use Microsoft Visual InterDev to create and manage a Web site. You will also learn how to create HTML pages and Active Server Pages by using the FrontPage Editor and the Visual InterDev Source Editor.

Chapter 3: Using Visual InterDev Data Tools

In this chapter, you will learn how to connect to and access a database from a Web page by working with the Data Form Wizard and the design-time data range controls in Visual InterDev. You will also learn how to use Visual InterDev DataView and the Query Designer to create SQL queries for a database.

Chapter 4: Using Objects on Web Pages

In this chapter, you will learn how to create interactive Web pages by adding ActiveX controls and Java applets to a Web page. You will also learn how Microsoft Internet Explorer downloads and runs these objects to exchange data with a Web server.

Chapter 5: Adding Client-Side Script

In this chapter, you will learn how to add client-side script to a Web page. You will learn how to use client-side script to create active Web pages that respond to user input, validate data in a form, and to manipulate objects, such as ActiveX controls and Java applets.

Chapter 6: Using Active Server Pages

In this chapter, you will learn how to use Active Server Pages in your Web application. You will learn how to read information from an HTTP request and customize the HTTP response, store information about a user, and determine the capabilities of a client's browser.

Chapter 7: Creating Database-Aware Web Pages

In this chapter, you will learn how to create Web pages that can retrieve and update information in a database. You will learn how to use ActiveX Data Objects (ADO) and the Advanced Data Connector (ADC) to connect to data sources and display data.

Chapter 8: Creating ActiveX Server Components

In this chapter, you will learn how to create business services that run on a Web server. You will learn how to use Visual Basic 5.0 to build ActiveX server components that contain business rules, and to call these components from a page on your Web site.

Chapter 9: Using Microsoft Transaction Server

In this chapter, you will learn about Microsoft Transaction Server (MTS), an application that provides transaction and resource management for ActiveX server components. You will also learn how to create Microsoft Transaction Server components, which are ActiveX server components that work within the MTS application environment.

Chapter 10: Controlling Access to a Web Site

In this chapter, you will learn how to use Windows NT and Internet Information Server (IIS) to control the access to your Web site. You will learn how to control which files a user can access, and which operations the user can perform on your Web server. You will also learn how to use digital certificates to identify users.

Lab Content

This *Mastering Web Site Development* course includes lab exercises at the end of each chapter. Each lab consists of two or more exercises that provide hands-on experience using the procedures and concepts you have learned in the associated chapter.

To complete the lab exercises, you must have the required software. For detailed information about the setup for the labs and the required software, see Labs.

Additional Course Information

You can use *Mastering Web Site Development* as more than a tutorial. It contains a variety of building blocks and tools that can assist you in creating solutions. For example, the Library section contains additional information that you may find useful as you work through the chapters.

In each chapter, the **Related Information** button provides jumps to specific chapters in this course, as well as to resources in the Library section.

In *Mastering Web Site Development*, you can use the Contents menu at the top of the Navigation pane to access the following types of information.

Information	Description
Labs	Step-by-step exercises that provide practice using the skills learned in each chapter.
Multimedia	Media elements that include expert point-of-view videos that discuss features, demonstrations that show how to perform tasks discussed in the text, and animations that illustrate a technical concept.
Self-Check Questions	Questions that test your understanding of the material in each chapter.
Articles and White Papers	This section of the Library includes a variety of technical articles and white papers.
Books	This section of the Library includes several online books and sample publications from Microsoft Press.
Resources	This section of the Library describes various Microsoft and third-party resources that you can refer to as you develop applications.
Knowledge Base	This section of the Library contains selected articles from the Microsoft Knowledge Base. The Microsoft Knowledge Base contains detailed technical articles and samples that are created and maintained by Microsoft Product Support Services (PSS).
Sample Applications	This section of the Library contains one of the sample applications provided on the Microsoft Visual InterDev Sample Applications Web page.

Return to Welcome Screen

Activate

Also Activation. A programming process that loads an object into memory, putting the object into an executable or running state. Also, the process of binding an object so as to put the object into its running state.

Active Client

The Active Client is the client-side element of the Active Platform that enables cross-platform content and applications. It includes support for HTML, scripting (VBScript and JScript), Java applets, ActiveX Components, ActiveX Controls, and Active Documents.

Active Document

A Windows-based, non-HTML application embedded in a browser, providing a way for the functionality of these applications to be accessible from within the browser interface.

Active Group, The

A standards organization, under the auspices of The Open Group, an open, customer-driven steering committee responsible for the ongoing development and management of ActiveX technologies and licensing.

Active Platform

An integrated, comprehensive set of client—Active Client—and server—Active Server—component-based development technologies that make it easy for developers to integrate the connectivity of the Internet with the power of the personal computer.

Active Server

The Active Server is the server-side element of the Active Platform, specifically, a collection of server-side technologies that are delivered with Windows NT, and provide a consistent server-side component and scripting model and an integrated set of system services for component application management, database access, transactions, and messaging.

Active Server Page

The server-side execution environment in Microsoft Internet Information Server 3.0 that executes ActiveX Scripts and ActiveX Components on a server.

ActiveX

A set of language-independent interoperability technologies that enable software components written in different languages to work together in networked environments. The core technology elements of ActiveX are COM and DCOM.

ActiveX Automation

A language-neutral way to manipulate an ActiveX Component's methods from outside an application. ActiveX Automation is typically used to create components that expose methods to programming tools and macro languages.

ActiveX Control

A compiled software component based on the component object model (COM) that encapsulates a set of business or user interface functions. An ActiveX Control is used to provide user interface components and is designed to run on the client computer.

ActiveX Component

A compiled software component based on the component object model (COM) that encapsulates a set of business functionality. The functionality in an ActiveX component is accessed through ActiveX Automation interfaces. The ActiveX Component can execute either on a client computer or on a server computer, transparent to the calling application, through DCOM.

ActiveX Server Component
An ActiveX Component designed to run on the server-side of a client/server application.

ActiveX Scripting
The act of using a scripting language to drive ActiveX Components.

ADO
ActiveX Data Objects. A set of object-based data access interfaces optimized for Internet-based, data-centric applications.

Aggregation

A programming composition technique for implementing component objects. Using this technique, developers can build a new object using one or more existing objects that support some or all of the new object's required interfaces.

Anonymous FTP

Anonymous File Transfer Protocol. Used in the process of connecting to a remote computer as an anonymous or guest user in order to transfer public files to your local computer.

ANSI

American National Standards Institute. ANSI serves as a quasi-national standards organization. It provides area charters for groups that establish standards in specific fields, such as the Institute of Electrical and Electronics Engineers (IEEE). Also, commonly used to refer to a low-level table of codes used by a computer.

Apartment Model Multi-threading

The Component Object Model (COM) supports a form of multi-threading in Windows 95 and Windows NT called the *apartment model.* Apartment is essentially a way of describing a thread with a message queue that supports COM objects.

API

Application Programming Interface. A set of routines that an application program uses to request and carry out lower-level services performed by a computer's operating system.

Applet
An HTML-based program built with Java that a browser temporarily downloads to a user's hard disk, from which location it runs when the Web page is open.

Asynchronous Call
A function that enables processing to continue without waiting for the function to return a value.

ATM
Asynchronous Transfer Mode. A communications protocol defined for high-speed data communications.

Automation

Bandwidth

The capacity of the transmission medium stated in bits per second (bps) or as a frequency (Hz). Generally, a higher bandwidth number indicates faster data-transfer capability.

Bind

Also Binding. To put an object into its running state, allowing the operations it supports to be invoked. Objects can be bound at run time—called *late binding* or *dynamic binding*—or at compile time—called *static binding*.

Browser
A program that interprets hypertext markup language (HTML) and displays information on a computer screen.

Bytecode
The executable form of Java code that executes within the Java Virtual Machine. Also called interpreted code, pseudocode, or p-code.

Cache
Usually a temporary local store for information, a special memory subsystem where frequently used data values are copied and stored for quick access.

Call

To transfer program execution to some other section of code, usually a subroutine, while saving the necessary information to allow execution to resume at the calling point when the called section has completed execution.

CASE

Computer Aided Software Engineering. Software that aids in application development including analysis, design, and code generation. CASE tools provide automated methods for designing and documenting traditional-structure programming techniques.

Certificate Authority

Certificate Authorities are companies that distribute certificates to software developers. To guarantee a control's authenticity, a Certificate Authority, such as the Verisign Corporation, develops a digital certificate for each developer who uses Authenticode technologies from Microsoft.

CGI

Common Gateway Interface. A server-side interface for initiating software services. A set of interfaces that describe how a Web server communicates with software on the same computer.

Class

A generalized category in object-oriented programming that describes a group of more specific items called objects. A class provides a template for defining the behavior of a particular type of object.

Class Identifier

Also CLASSID or CLSID. A unique identification tag (UUID) associated with a class object. A class object that is intended to create more than one object registers its CLSID in a task table in the system registration database to enable clients to locate and load the executable code associated with the object(s).

Class Library
A collection of one or more classes that programmers use to implement functionality.

Class Object
A member object within a class.

Client

A program that facilitates a connection to server computers and manages and presents information retrieved from those sources. In a client/server environment, the workstation is usually the client computer. In referring to COM objects, an object that requests services from another object.

Client/Server

A model of computing whereby client applications running on a desktop or personal computer access information on remote servers or host computers.

COM

Component Object Model. The object-oriented programming model that defines how objects interact within a single application or between applications.

Communications Protocol

A set of rules or standards designed to enable computers to connect with one another and to exchange information with as few errors as possible.

Component

Compound Document
A document that contains data in different formats created by different applications.

Container Application

A container application provides storage for the embedded object, a site for display, access to the display site, and an advisory sink for receiving notification of changes in the object.

Control
In a graphical user interface, an object on the screen that can be manipulated by a user to perform an action.

Cookies

A means by which, under the HTTP protocol, a server or a script can maintain state or *status* information on the client workstation.

CORBA
Common Object Request Broker Architecture. An Object Management Group specification for the interface definition between OMG-compliant objects.

Cursor Engine
A mechanism for managing data retrieved from a database, or a full transaction manager that optimizes the retrieval and update of server-based data.

DAO
Data Access Objects. DAO includes the full functionality of the Microsoft Jet database engine for local data management.

Data Dictionary

A repository of information about data, such as its meaning, relationships to other data, origin, usage, and format.

DCE
Distributed Computing Environment. An open set of services controlled by the OSF and designed to support performing distributed computing across heterogeneous platforms.

DCOM

Distributed Component Object Model. Additions to the Component Object Model (COM) that facilitate the transparent distribution of objects over networks and over the Internet.

Debugger
A development environment that supports step-by-step execution of application code and viewing the content of code variables.

Design-time ActiveX Controls

Visual authoring components that help a developer construct dynamic Web applications by automatically generating standard HTML and/or scripting code. They are analogous to wizards.

Distributed Processing

The physical or logical distribution of software components, processing, data, and management of application software.

DNS

Domain Name Service. A protocol that provides an Internet-wide database of host and domain names. For example, DNS is used to find the IP address of a host name written as *microsoft.com*.

Domain Name
An entry in an Internet address, such as *microsoft.com* in the fictitious U.S. address www.example.microsoft.com/.

E-commerce
Electronic Commerce. The process of buying and selling over the Web—often based on software products such as the Microsoft Merchant Server.

Event
Any action, often generated by a user or an ActiveX Control, to which a program might respond.

FAQ
Frequently Asked Questions. Usually a document containing questions and answers that address the basics.

Firewall
A security mechanism—such as the Microsoft Proxy Server—that provides Internet access from desktops inside an organization, while at the same time preventing access to the corporate LAN by outside Internet users.

FTP
File Transfer Protocol. The Internet standard high-speed protocol for downloading or transferring files from one computer to another.

Function
A general term used for a subroutine. In some programming languages, a subroutine or statement that returns values.

GIF

Graphics Interchange Format. A computer graphics file format developed in the mid-1980s by CompuServe for use in photo-quality graphic image display on computer screens.

Gopher
An early Internet protocol and software program designed to search for, retrieve, and display documents from remote computers or sites.

GUI

Graphical User Interface. A user interface that displays graphics and characters and provides an event model for users to control the operating environment.

GUID

Globally Unique Identifier. Identifiers (IDs) assigned to COM objects that are generated through a sophisticated algorithm. The algorithm guarantees that all COM objects are assigned unique IDs, avoiding any possibility of a naming conflict.

Home Page
The page that serves as the starting point of a World Wide Web site, sometimes named *default.html* or *index.html*.

Host
Any computer that provides services to remote computers or users.

HTML

Hypertext Markup Language. A tag-based notation language used to format documents that can then be interpreted and rendered by an Internet browser.

HTTP
Hypertext Transfer Protocol. A basic communication protocol for Internet or Web server file input and output (I/O).

Hyperlink
A connection to a document or other file on the Internet that generally appears as a highlighted word or image on the screen.

Hypertext

A hypertext document is a document that is structured in chunks of text, marked up (usually using HTML), and connected by links. Hence, the text in the document can properly be named hypertext because of its marked-up and navigable condition.

IDC
Internet Database Connector. Provides database connectivity between IIS applications and any ODBC-compliant database.

IEEE
Institute of Electrical and Electronic Engineers.

IETF
Internet Engineering Task Force. A protocol engineering and development organization focused on the Internet.

IIS
Microsoft Internet Information Server.

Inheritance
A programming technique that duplicates the characteristics down a hierarchy from one class to another.

In-process Server
An ActiveX Component that shares the same memory as the container application.

Instance
An object for which memory is allocated or persistent.

Instantiate
To create an instance of an object. The process of creating or activating an object based on its class.

Interface
A group of related functions that provide access to COM objects.

Internet

Abbreviation for Internetwork. A set of dissimilar computer networks joined together by means of gateways that handle data transfer and the conversion of messages from the sending network to the protocols used by the receiving networks.

Intranet

Use of Internet standards, technologies, and products within an enterprise to function as a collaborative processing infrastructure. The term intranet is generally used to describe the application of Internet technologies on internal corporate networks.

IP
Internet Protocol. The packet-switching protocol for network communications between Internet host computers.

ISAM
Indexed Sequential Access Method. An indexing mechanism for efficient access to rows of data in a file.

ISAPI
Internet Server Application Program Interface. An application program interface that resides on a server computer for initiating software services tuned for Microsoft Windows NT operating system.

ISDN
Integrated Services Digital Network. An emerging technology that is beginning to be offered by most telephone service providers as a faster alternative to traditional modems.

ISO
International Standards Organization**.** An organization involved in setting standards worldwide for all fields except electro-technical, which is the responsibility of IEC.

ISP
Internet Service Provider. An organization that provides access to the Internet.

ISV
Independent Software Vendor.

ITU
International Telecommunication Union.

Java

A derivative of the C++ language, Java is Sun Microsystems Corporation distributed programming language, offered as an open standard.

JavaScript
A scripting language that evolved from Netscape's LiveScript language and was made more compatible with Java. It uses an HTML page as its interface.

Java Beans

An object model being developed by Sun Microsystems Corporation that is targeted to inter-operate with a variety of other object models, including COM and CORBA.

JDBC
Java Database Connectivity. Data access interfaces based on ODBC for use with the Java language.

Jet
A Microsoft desktop database engine available in most of Microsoft's development tools and office products, including Microsoft Access, Microsoft Office, and Microsoft Visual Basic.

JPEG
Joint Photographic Experts Group. A widely accepted international standard for compression of color image files, sometimes used on the Internet.

JScript

The Microsoft open implementation of JavaScript. JScript is fully compatible with JavaScript in Netscape Navigator version 2.0.

Kerberos

The basis of most of the distributed computing environment (DCE) security services. Kerberos provides the secure use of distributed software components.

Latency

The state of being latent, or to lie hidden; not currently showing signs of existence. Sometimes attributed to the time taken to retrieve pages from the World Wide Web.

LAN

Local Area Network. A connection among a set of computers. Computers connected to a LAN can generally share applications or files from a local file server and may be able to connect to other LANs or to the Internet using routers.

LDAP

Lightweight Directory Access Protocol. An open standard protocol (RFC 1777) that provides a way for Internet clients, applications, and servers to access directory services. LDAP was derived from the DAP X.500 protocol.

Link

Marshal

Also Marshalling. The process of packaging and sending interface parameters across process boundaries in computer memory.

Message Queuing

Server technology developers can use to build large-scale distributed systems with reliable communications between applications that can continue to operate reliably even when networked systems are unavailable.

Method
Member functions of an exposed object that perform some action on an object, such as saving it to disk.

MIME

Multipurpose Internet Mail Extensions. An extension of the Internet mail protocol that enables users to send 8-bit based e-mail messages, which are used to support extended character sets, voice mail, facsimile images, and so forth.

Moniker
A name that uniquely identifies a COM object, similar to a directory path name.

MTS
Microsoft Transaction Server. Combines the features of a transaction-processing (TP) monitor and an object-request broker (ORB) in an easy-to-use product.

Multi-tasking
The ability to simultaneously execute multiple applications within an operating system.

Multi-tier Architecture

Also known as three-tier, multi-tier is a technique for building applications generally split into user, business, and data services tiers. These applications are built of component services that are based on an object model such as ActiveX.

Multi-threading

Running several processes in rapid sequence within a single program, regardless of which logical method of multi-tasking is being used by the operating system.

NIST
National Institute of Standards and Technology.

NNTP

Network News Transport Protocol. A news service protocol that is an extension of the TCP/IP local area network protocol. NNTP is the standard for Internet exchange of Usenet messages.

Node
A computer that is attached to a network; also called a *host.* Also, a junction of some kind. On a local area network, a device that is connected to the network and is capable of communicating with other network devices.

Object
In object-oriented programming, a variable comprising both routines and data that is treated as a discrete entity.

OCX

File extension for an ActiveX Control or ActiveX Component. Originally used as a file extension for OLE Custom Controls, following the format for a Visual Basic Extension (VBX).

ODBC
Open Database Connectivity. A developer can use ODBC to access data in a heterogeneous environment of relational and non-relational databases.

ODBCDirect
Technology that makes the full functionality of RDO available from within DAO. Used to bypass the Microsoft Jet database engine for fast, small-memory-footprint access to remote data.

OLAP
Online Analytical Processing. A multi-dimensional database used for decision support analysis and data warehousing.

OLE
Object Linking and Embedding. A set of integration standards to transfer and share information among client applications.

OLE Automation

OLE Control

OLE DB

Data-access interfaces providing consistent access to SQL and non-SQL data sources across the enterprise and the Internet.

OMG
Object Management Group. A vendor alliance formed to define and promote CORBA object specifications.

Open Group, The

Parent company of a number of standards organizations, including The Active Group—now managing the core ActiveX technology, X/Open, and OSF.

OSF
Open Software Foundation. A vendor alliance to define specifications, develop software, and make available an open, portable environment. Now merged with The Open Group.

ORB
Object Request Broker. Manages interaction between clients and servers including the distributed computing responsibilities of location referencing as well as coordinating parameters and results.

Out-of-Process
An ActiveX Component that runs in its own separate memory space separate from a container application.

PCT
Private Communication Technology. Designed to provide secure transactions over the Internet.

PKCS

Public Key Certificate Standard. Syntax standards covering a number of security functions, including a standard way of attaching signatures to a block of data, a form for requesting a certificate, and public key encryption algorithms.

POP3

Post Office Protocol version 3. A messaging protocol commonly used on the Internet. It stores and forwards e-mail to users that log on to the mail server. POP uses the SMTP message format protocol.

PPP

Point-to-Point Protocol. The Internet standard for serial communications, PPP defines how data packets are exchanged with other Internet-based systems using a modem connection.

PPTP

Point-to-Point Tunneling Protocol. The Internet can be used for low-cost, secure remote access to a corporate network with virtual private networking support on Windows NT.

progID
A string expression that is the programmatic ID of the new object in a component.

Property
A set of characteristics of an object.

Proxy Server

A proxy server acts as a go-between, converting information from Web servers into HTML to be delivered to a client computer. It also provides a way to deliver network services to computers on a secure subnet without those computers needing to have direct access to the World Wide Web.

Protocol
A mutually determined set of formats and procedures for the exchange of information between computers.

RAD
Rapid Application Development.

RDO
Remote Data Objects. In version 2.0, RDO is a high-level object interface that directly calls ODBC for optimal speed, control, and ease of programming.

Router

An intermediary device on a communications network responsible for deciding by which of several paths message traffic will flow over a network or the Internet.

RPC

Remote Procedure Call. A mechanism that extends the notion of a local procedure call—meaning contained in a single memory address space—to a distributed computing environment.

RSA

A public key cryptography for Internet security. This acronym derives from the last names of the inventors of the technology: Rivest, Shamir, and Adleman.

RTP/RTCP

Real-time protocol and real-time control protocol, respectively. A packet format for sending real-time information across the Internet.

Scalability
The capability to use the same software environment on many classes of computers and hardware configurations.

SET

Secure Electronic Transactions. A protocol for securing electronic credit card payments when conducting commerce across the Internet, which is broadly supported by companies in the computer and banking industries —including MasterCard and Visa.

Script
A kind of program that consists of a set of instructions for an application or utility program.

SDK
Software Development Kit.

SEPP
Secure Electronic Payment Process. A proposed specification that merged with STT, resulting in the SET standard for secure e-commerce transactions.

Server
A computer-running administrative software that controls access to all or part of a network and its resources.

SGML

Standard Generalized Markup Language. An original documentation markup standard promulgated by primary defense contractors as a standard for the development and display of documentation. HTML is a subset of SGML.

SMP

Symmetric Multiprocessing. A multiprocessor architecture in which all processors are identical, share memory, and execute both user code and operating system code.

SMTP
Simple Mail Transfer Protocol. The Internet standard protocol for transferring electronic mail messages from one computer to another.

SQL
Structured Query Language. The international standard language for defining and accessing relational databases.

SQL Access Group (SAG)
A consortium of vendors established in November 1989 to accelerate the Remote Data Access standard and to deliver protocols for interconnectivity among multiple SQL-based software products.

SSL

Secure Sockets Layer. A standard for providing encrypted and authenticated service over the Internet. Uses RSA public-key encryption for specific TCP/IP ports.

Stored Procedures

Pre-compiled software functions that are managed and that run within a remote database management system (RDBMS).

STT

Secure Transaction Technology. A proposed specification that merged with SEPP, resulting in the SET standard for secure e-commerce transactions.

Synchronous
A function that does not allow further instructions in the process—code—to be executed until the function returns a value.

TCP/IP

Transmission Control Protocol/Internet Protocol. TCP/IP is a combined set of protocols that perform the transfer of data between two computers.

Telnet

A terminal emulation protocol users can employ to log on to other computers on the Internet. Alternatively, software that can be used to log on to another computer using the telnet protocol.

Three-tier Architecture

Transaction

A group of processing activities that are either entirely completed, or if not completed, that leave the database and processing system in the same state as before the transaction started.

TP

Transaction Processing. The real-time handling of computerized business transactions as they are received by the system. Also called online transaction processing (OLTP) systems.

Two-tier Architecture

URL

Uniform Resource Locator. An address that uniquely identifies a World Wide Web site, usually preceded with *http://* such as in this fictitious URL *http://www.example.microsoft.com/.* A URL can contain more detail, such as the name of a page of hypertext, usually identified by the suffix .html or .htm.

VBA
Visual Basic, Applications Edition. The development environment and language found in Visual Basic that can be hosted by applications.

VBX
Visual Basic Extension. Custom controls originally designed for 16-bit applications created by Visual Basic.

Virtual Machine

The mechanism the Java language uses to execute Java bytecode on any physical computer. The VM converts the bytecode to the native instruction for the target computer.

Virtual Root

Also Vroot. A virtual tree of Web aliases that points to local, physical directories. This simplifies client URL addresses by presenting an entire set of content directories as a single directory tree.

VRML
Virtual Reality Modeling Language. A language for coding three-dimensional HTML applications.

W3C
World Wide Web Consortium.

Web Application

Also Web-based Application. A software program that uses HTTP for its core communication protocol and delivers Web-based information to the user in the HTML language.

Windows Sockets

Also Winsock. Winsock provides a single interface in Microsoft Windows to which multiple network software programs conform.

World Wide Web

Also the Web or WWW. The Web is a collection of Internet host systems that make these services available on the Internet using the HTTP protocol. Web-based information is usually delivered in the form of hypertext and hypermedia using HTML.

WOSA

Windows Open Services Architecture. An architecture and set of application programming interfaces for Windows that standardized the interfaces developers use in accessing underlying network services.

WYSIWYG

What You See Is What You Get. Authoring software programs that render a document on the computer screen the way it will appear in print, even as it is being edited.

XA

A transaction interoperability standard defined by X/Open. The Microsoft Transaction Server uses XA to connect with other transaction processing systems.

X.500 (including DAP)
Directory Access Protocol is a standard for global directory services.

X.509 Certificate
A protocol for a cryptographic certificate that contains a vendor’s unique name and the vendor’s public key.

X/Open
An independent consortium of international computer vendors created to establish multi-vendor standards based on *de facto* and *de jure* standards.

Click here to connect to the Knowledge Base page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The Microsoft Knowledge Base contains detailed technical articles and samples that are created and maintained by Microsoft Product Support Services (PSS). This comprehensive database contains more than 50,000 detailed articles with technical information about Microsoft products, fix lists, documentation errors, and answers to commonly asked technical support questions. The Knowledge Base is updated daily and is a primary Microsoft product information source used by Microsoft support engineers every day.

This section contains the following articles:

Note To help you quickly locate articles, we have sorted the table of contents (in the Navigation pane on the left) in alphabetical order, based on article title, and the following bullet list of articles in numerical order, based on article number.

For search hints and information on categories and keywords in the Knowledge Base, check out Q94671: Categories and Keywords for All Knowledge Base Articles.

® Q85774: Instructions for Using the Microsoft Download Service (MSDL)
® Q94671: Categories and Key Words for All Knowledge Base Articles
® Q119591: How to Obtain Microsoft Support Files From Online Services
® Q142868: Authentication and Security Features
® Q143090: FrontPage: Overview of FrontPage WebBot Components
® Q143130: Microsoft Transaction Server FAQs
® Q149054: Choosing a rdoResultset Cursortype
® Q152828: IIS Queries to SQL Server Generate Error 1326
® Q157748: How to Modify .alx File Objects From Active Server Pages
® Q157959: How to Package MFC Controls for Use Over the Internet
® Q158737: How to Create a Simple Query in an ActiveX Layout
® Q159325: Server and Browser Requirements for Publish to Web Wizard
® Q159402: How to Use Response.Redirect in a Server Script
® Q159682: "Data Source Name Not Found" Err Msg Opening Web Page
® Q159976: How to Connect to the Microsoft SQL Server Via Named Pages
® Q159977: How to Stop Users From Displaying a Frame Outside Its Frameset
® Q160226: Can't Use Proofing Tools (Spelling Checker) or Text Converters
® Q160682: FAQs About Microsoft Merchant Server
® Q160754: Error "HTTP/1.0 403 Access Forbidden" Browsing IDC Page
® Q160833: How FrontPage 97 Handles Document Conversion to HTML
® Q161172: How to Use IDC Files to Query a Secure MS Access Database
® Q161333: Check NT Permissions When Using IDC/ASP Files With Access
® Q161420: FrontPage Editor Deletes Unknown Attributes in HTML
® Q161779: Using FrontPage 97 to Edit, Manage Active Server Pages
® Q162145: FrontPage Configuration Settings for Windows NT Servers
® Q162840: ADO.Connection Updated to ADODB.Connection
® Q162908: Lookup Fields Ignored When Exporting to Internet Formats
® Q162975: Permissions Necessary to View HTML, IDC, and ASP Files
® Q162976: How to Verify That ASP is Working on Your Web Server
® Q162977: ASP Query Cannot Be Used With the LIKE Predicate
® Q162981: Export to ASP or IDC Ignores Filter/OrderBy Properties

® Q163009: Values for Scripting Object Constants Defined
® Q163010: Disabling Cookies Sent By Active Server Pages
® Q163014: Format Properties Ignored When Exporting Queries to ASP
® Q163034: Form Control Format Property Ignored When Exported to ASP
® Q163133: Searching for ASP Articles by KBSubcategory
® Q163159: How to Use ASP Files to Query a Secure MS Access Database
® Q163443: Action Queries Cannot Be Exported to HTML, IDC, or ASP
® Q163485: Active Server Pages Script Appears in Browser
® Q163499: Creating a Dynamically Growing Form Using ASP
® Q163501: INVALID APPLICATION NAME Error in Active Server Pages
® Q163510: Constant for OutputTo Method Incorrect in Help
® Q163603: Files for Testing System DSN Available on MSL
® Q163706: ASP Files Display Hyperlinks as Text in Web Browser
® Q163948: Data in ASP Fields With Spaces Not Committed to Access DB
® Q164001: How to Download Active Server Pages From the Internet
® Q164002: Web Browser Error Opening Subform in ASP File
® Q164004: MS Internet Explorer on High Safety Stops Subform Display
® Q164008: Subforms Appear Blank When Browsing ASP Files on NT 4.0
® Q164059: Execution File Text Can Be Viewed in Client
® Q164159: How to Use ASP Files to Query a Secure MS Access Database
® Q164586: How to Use OLE DB Sample Text Provider in ADO
® Q165293: Declaring an Array at Application Level Scope
® Q165831: Anonymous User in NT Admin Group Breaks Source Control

The *Mastering Web Site Development* course contains lab exercises that give you experience using the information presented in the chapters.

Lab Scenario

The scenario used for all labs is a Web site for a fictional university called State University. In this Web site, university students can retrieve information about classes, enroll in classes, and receive customized transcripts of their grades.

In Lab 2: Developing a Web Project, you create a Web project with a home page. In Lab 3: Using Visual InterDev Data Tools, you build the database used by the Web site. In subsequent labs, you add and modify Web pages to refine the Web site.

The following table lists the tasks you perform with the fictitious Web site, and provides a description of the lab in which you will accomplish the task.

Task	Lab description
View the home page of the StateU Web site.	In Lab 2: Developing a Web Project, you create the home page for the StateU Web site. The home page displays links to other pages on the site.
View class information.	In Lab 3: Using Visual InterDev Data Tools, you use the Data Form Wizard and design-time controls to create Web pages that display class information.
Display a list of classes required for a particular major.	In Lab 4: Using Objects on a Web Page, you add an ActiveX control pop-up menu to the StateU home page to display a list of classes required for each major.
View the State University mascot.	In Lab 4, you also add a Java applet that animates the State University mascot.
Create a profile form.	In Lab 5: Adding Client-Side Script, you create a profile form for students that records their ID, name, and major when they first access the Web site.
	In Lab 6: Using Active Server Pages, you add server-side script that uses the information in the profile form to customize Web pages returned to each student.
Retrieve a student transcript.	In Lab 7.1: Using ActiveX Data Objects, you add a Web page that reads information from the database and displays a transcript for a student.
Display a list of classes.	In Lab 7.2: Using the Advanced Data Connector, you add a Web page that uses ADC to display a dynamic list of classes.
Add, drop, and transfer classes.	In Lab 8: Creating ActiveX Server Components, you use Visual Basic 5.0 to create business objects that run on a Web server. These objects are run when students add, drop, or transfer classes.
	In Lab 9: Using Microsoft Transaction Server, you use MTS to manage business objects.
Control access to a Web site, pages, and files.	In Lab 10: Controlling Access to a Web Site, you prevent anonymous logons and set permissions for a Web site and specific Web pages.

Lab Setup

To complete the labs, you need the following software and resources:

- Access to a Web server. You can use one of the following options:
  - Microsoft Internet Information Server (IIS) version 3.0 on Windows NT Server version 4.0.

This is the recommended setup, and it is required to complete Lab 9.

- Microsoft Peer Web Services on Microsoft Windows NT workstation version 4.0.
- Microsoft Windows 95 Personal Web Server on Microsoft Windows 95.

® Microsoft Visual InterDev 1.0 server components installed on your Web server. You need to install the following server components:

- Active Server Pages

  For information about installing Active Server Pages, see the technical white paper, Active Server Pages Installation Notes in the Articles and White Papers section of the Library.

- FrontPage Server Extensions

® Microsoft Visual InterDev 1.0 client components installed on your development computer.

The only client component that is required for this course is Visual InterDev 1.0.

For information about installing Visual InterDev, see the technical white paper, Microsoft Visual InterDev Installation Notes in the Articles and White Papers section of the Library.

® Lab database

Two versions of the lab database are supplied with this course: a Microsoft Access version and a Microsoft SQL Server version. You can use either version for Labs 2—8 and Lab 10.

To complete Lab 9, you must use the SQL Server version of the database, with Microsoft SQL Server version 6.5 or later.

® Microsoft Visual Basic version 5.0

To complete Lab 8, you must have Microsoft Visual Basic version 5.0 Professional Edition.

® Microsoft Transaction Server

To complete Lab 9, you must have Microsoft Transaction Server version 1.0 or later.

For information about installing Microsoft Transaction Server, see the technical white paper, Microsoft Transaction Server Installation Notes in the Articles and White Papers section of the Library.

Installing Lab Code

For each lab, there is associated code, as well as project and solution files. If you installed the labs during Setup, these files are in the folder *<Install Folder>*\Labs\Lab*xx* on your hard disk. If you did not install the labs during Setup, you can find them in the folder \Labs\Lab*xx* on the *Mastering Web Site Development* CD-ROM. To install the lab files, re-install the course and choose the Complete installation option.

{ewc MVIMG, MVIMAGE,!library.shg}

Mastering Web Site Development provides numerous audio/video demonstrations, animations, and expert points of view that illustrate the concepts and techniques discussed in this course.

When you see this icon {ewc MVIMG, MVIMAGE,!democlip.bmp}, click it to launch a demonstration of the topic being discussed.

When you see this icon {ewc MVIMG, MVIMAGE,!anim.bmp}, click it to launch an animation on the topic.

When you see this icon {ewc MVIMG, MVIMAGE,!exppov.bmp}, click it to launch an expert point of view.

Note You can turn the display of text in a demonstration or animation on and off by clicking the **CC** (Closed Caption) button when viewing the media piece.

This section provides an overview of how to use the available scripting languages.

The main purpose of adding script to a Web page is to create event procedures for objects on a page, such as ActiveX controls and standard HTML controls.

This section begins by describing the differences between client-side script and server-side script. You will then learn how to use HTML tags to name an object so you can create event procedures for the object. You will also learn how to use the HTML object model to reference objects on a Web page.

This section includes the following topics:

Scripting is code that you write in a Web page. The script runs either on the client computer when a user interacts with a control, or on the Web server before the page is returned to the client. In both cases, you add script to a Web page as ASCII text.

For information about coding conventions, search on "Coding Conventions" in the Visual InterDev InfoViewer.

Client-Side Script

Client-side script runs on the client computer. Web browsers contain scripting interpreters that can read and run the script.

The primary purpose of adding client-side script to a Web page is to create event procedures for controls. For example, you can write an event procedure that runs a function when a user clicks a particular button.

Client-side script is not compiled, nor is it encrypted on an HTML page. Therefore, if users view the HTML source of a Web page, they will see the script included in the page.

Client-side script requires a Web browser to support the scripting language. If a Web browser does not support the scripting language, the user will not have full access to the features on the Web page.

Server-Side Script

Server-side script runs in an Active Server Page on a Web server before the server returns the page to the user. When a user requests an Active Server Page, the server-side script runs and generates HTML to return to the user. Server-side script is not visible to the user on the returned Web page.

Because server-side script runs on a Web server, the script has access to all resources that reside on that server, such as databases and executable files.

Server-side script requires a Web server to support Active Server Pages. Active Server Pages is a Microsoft Internet Server Application Program Interface (ISAPI) technology. A Web browser does not need to provide any additional functionality. Therefore, server-side script will run regardless of which Web browser is used.

Common Tasks of Client and Server Script

When you use client-side script, processing occurs on the user's computer while the user interacts with the Web page.

Client-side script is useful for performing the following tasks:

® Validating control values on a form

® Providing event procedures for controls

When you use server-side script, processing occurs on the Web server before the page is returned to the user.

Server-side script is useful for performing the following tasks:

® Accessing a database and returning data to the user

® Saving status information about a user or session

Visual InterDev and FrontPage both include the Script Wizard. You can use the Script Wizard to assign actions to controls and add VBScript or JavaScript to a Web page.

To see a demonstration of how to use the Script Wizard, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Viewing Properties and Methods

You can use the Script Wizard to view the properties and methods for any HTML or ActiveX control on a Web page. When you start the Script Wizard, the controls and their events are listed in the left pane, and the control properties and methods are listed in the right pane.

To see an illustration of the Script Wizard with events, properties, and methods displayed for controls, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Setting the Language

The Script Wizard can generate either VBScript or JavaScript. To set the language that the Script Wizard will use, click **Options** on the **Tools** menu in Visual InterDev. On the **HTML** tab, select **VBScript** or **JScript** as the default language for the Script Wizard.

When you run the Script Wizard, it displays the script that is currently in your Web page. Although you can have both JavaScript and VBScript in the same page, the Script Wizard will recognize only the script written in the scripting language that is currently set for the Script Wizard.

Starting the Script Wizard

u **To start the Script Wizard in Visual InterDev**

1. Open the Web page to which you want to add the script.
2. Right-click anywhere in the page, and then click **Script Wizard**.

u **To start the Script Wizard in FrontPage**

1. On the **Insert** menu, click **Script**.
2. In the **Script** dialog box, click **Script Wizard**.

Generating Script

To generate script, the Script Wizard provides two views: List view and Code view. To switch between views, select the appropriate option at the bottom of the Script Wizard.

® List view

In List view, you can define the actions you want to occur for the event procedures of objects. You first select the event procedure in the left pane, and then select the actions you want to occur in the right pane. The Script Wizard generates the script to perform the actions.

To see an illustration of the Script Wizard in List view with several actions selected for an event procedure, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

® Code view

In Code view, you can add script manually for any event procedure or user-defined procedure.

To add script to an event procedure, select it in the right pane, and type the script in the lower pane. To add script to a user-defined procedure, right-click the procedure name in the right pane, click **Edit**, and then type script in the lower pane.

To see an illustration of the Script Wizard in Code view, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

For more information about using the Script Wizard, search on "Script Wizard" in the Visual InterDev InfoViewer.

In this section, you will learn how to add client-side script to a Web page.

First, you will learn how to add the <SCRIPT> tag to define a script section on a Web page. In the script section, you will create event procedures and add code that invokes methods and accesses properties of objects.

You will also learn how to run script that initializes a Web page and validates data on a form. Finally, you will use the Visual InterDev Script Wizard to view the properties and methods of standard HTML and ActiveX controls on a Web page, and to generate script.

This section describes how to write script with VBScript and JavaScript, but focuses primarily on VBScript.

This section includes the following topics:

To add script to a Web page, you first define a <SCRIPT> section. In the <SCRIPT> section, you can write script that runs when the Web page is downloaded, when the script is explicitly invoked, or when an event occurs.

To see a demonstration of how to add VBScript code to an HTML page, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

<SCRIPT> Tag Syntax

To define a <SCRIPT> section, you insert the HTML <SCRIPT> tag. To specify which interpreter the browser should use to run the script, you set the LANGUAGE attribute of the <SCRIPT> tag.

For VBScript and JavaScript, you set the LANGUAGE attribute to VBScript or JavaScript, respectively. Some browsers, such as Microsoft Internet Explorer, include more than one scripting interpreter. If you do not specify the language, all Web browsers will use a JavaScript interpreter to read the script on the page.

The following table shows VBScript and JavaScript code that defines a procedure to display a message box.

VBScript

```
<SCRIPT LANGUAGE=VBScript>
<!--
Sub SayHello()
    MsgBox "Hello, world!"
End Sub
-->
</SCRIPT>
```

JavaScript

```
<SCRIPT LANGUAGE=JavaScript>
<!--
function SayHello()
{
    alert ("Hello, world!");
}
//-->
</SCRIPT>
```

Note Browsers that do not support scripting code will display the code as text in a Web page. To prevent browsers from displaying code as text, add comment tags (<!-- and -->) around the script code.

You can place script code anywhere on a Web page, but to simplify code maintenance, place all of your code within the same <SCRIPT> section. You can insert the <SCRIPT> tag in either the <BODY> or <HEAD> sections of the HTML page.

Running Script

In a <SCRIPT> section of a Web page, the location of where you add script will determine when the script will run.

In general, you can add script in one of the following three areas:

® Inline

If you add script outside of a procedure, the script will run when the browser encounters it as the page downloads. This is useful if you want to initialize data or objects on the page.

® Procedures

If you add script in a procedure, the script will run when the procedure is explicitly invoked.

® Event procedure

If you add script in an event procedure, the script will run when the event occurs. For example, if you create an OnClick event procedure for a button, the script will run when the user clicks the button.

To use an object in client-side script, you must first create the object, and then assign a name to it. You use this object name to create event procedures and to access the properties and methods of the object. The syntax for assigning names varies slightly for different types of objects.

For information about placing objects on a Web page, see Chapter 4: Using Objects on Web Pages.

Standard HTML Controls

To assign a name to a standard HTML control, you set the NAME attribute.

The following example code assigns the name cmdValidateOrder to a **Button** control:

```
<INPUT TYPE="BUTTON" NAME="cmdValidateOrder" VALUE="Validate Order">
```

ActiveX Controls

To assign a name to an ActiveX control, you set the ID attribute of the <OBJECT> tag.

The following example code assigns the name lblOccupation to an ActiveX **Label** control.

```
<OBJECT
    classid="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
    id=lblOccupation
>
```

Java Applets

To assign a name to a Java applet, you set the NAME attribute of the <APPLET> tag.

The following example code assigns the name outline to the Java Outline applet:

```
<APPLET
    CODE=Outline.class
    NAME=outline
    HEIGHT=150
    WIDTH=200>
</APPLET>
```

After you create and name the objects on a Web page, you can create event procedures for them. Script in the event procedure will run when the event occurs.

Common Events

Each object type has a set of events that it recognizes. Common events include clicking a control, entering text in a control, and moving the cursor over a control.

ActiveX controls and Java applets support the same events on an Web page as they do in other environments, such as Visual Basic or Visual J++.

To view the events supported by standard HTML controls and ActiveX controls, right-click the Web page in Visual InterDev, and then click **Script Wizard**. The Script Wizard displays the events supported by all standard HTML controls and ActiveX controls on the page.

For more information about the Script Wizard, see Using the Script Wizard.

Note Currently, Java applets do not trigger events, so event procedures cannot be written for them.

Creating an Event Procedure

There are a number of different ways to create an event procedure for an object:

® Name a procedure ObjectName_Event.

® Create a separate <SCRIPT> section for the event procedure.

® Assign the event procedure in an HTML tag for the object.

® Include the script in the HTML tag that defines the object.

Name the Procedure ObjectName_Event

If you name a procedure ObjectName_Event, the procedure will run automatically when the event for the object occurs. This naming convention is the same as the convention used to define event procedures in Visual Basic. This method is supported by VBScript only.

In the following example code, the procedure runs when the user clicks Button1:

```
Sub Button1_OnClick ()

End Sub
```

To see a demonstration of how to add an event procedure to a control by naming a procedure ObjectName_Event, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Create a Separate <SCRIPT> Section

You can create a separate <SCRIPT> section that contains script to run for a specific event of a control.

The following example code shows script that will run when the Click event of the **Calendar1** control occurs:

```
<SCRIPT LANGUAGE="JavaScript" FOR="Calendar1" EVENT="Click()">
<!--
    alert ("hello")
//-->
</SCRIPT>
```

You can use this method in either JavaScript or VBScript to create event procedures for ActiveX controls and standard HTML controls.

Assign the Event Procedure When Creating the Object

In the HTML tag that creates an object, you can specify an event name and the procedure to be invoked when that event occurs. You must include a <SCRIPT> section that includes the procedure declaration before the HTML tag that defines the object.

This method is useful if you want events from different objects to invoke the same procedure. You can use this method to assign event procedures for standard HTML controls.

In the following example code, the ProcessOrder procedure is called when the user clicks the radio button.

```
<SCRIPT LANGUAGE=VBScript>
Sub ProcessOrder ()
...
End Sub
</SCRIPT>
<INPUT TYPE=RADIO NAME=RadioGroup onClick="ProcessOrder">
```

This method is supported by both VBScript and JavaScript.

Include Script in the HTML Tag

In the HTML tag that creates the object, you can specify an event name and the script to run when that event occurs. You can use this method to assign event procedures for standard HTML controls.

In the following example code, when a user clicks the **Hello** button, the message "Hello World" will be displayed:

```
<INPUT LANGUAGE="VBScript" TYPE=button VALUE="hello" onClicK="Msgbox &quot;Hello
World&quot;">
```

This method is supported by JavaScript and VBScript.

The most common tasks performed by the event procedures of objects is accessing properties and invoking methods.

Accessing Properties

To access the value of a property, you use the following syntax:

Object.Property = Value

To retrieve the values of properties, you use this syntax:

ReturnValue = Object.Property

The following example code reads the date selected in a **Calendar** control, computes the shipping cost to deliver on that date, and then displays the calculated cost in a standard HTML text box:

```
Sub ShipDate_Click()
    Dim shiptime
    shiptime = ShipDate.Value - Date
    If shiptime < 0 Then
        MsgBox "Can't ship before today"
    ElseIf shiptime < 5 Then
        frmShip.txtCost.Value = 12
    Else
        frmShip.txtCost.Value = 8
    End If
End Sub
```

To see an illustration of the page, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Invoking Methods

You invoke methods in script code the same way you do in Visual Basic, by using the following syntax:

```
[Call] Object.Method
```

The following example code calls the **Today** method of the **Calendar** control when a user clicks the **Reset** button:

```
Sub Reset_OnClick()
    call ShipDate.Today()
End Sub
..
```

The **Call** keyword is optional. If you specify **Call**, you must enclose the arguments in parentheses.

You can write client-side script that runs when the browser first loads the Web page. This script can initialize variables and set properties of the controls and other objects on the page.

There are two ways in which you can write initialization script:

® Place script within a <SCRIPT> tag outside of a procedure.

® Add an OnLoad event procedure for the page.

The initialization script within a <SCRIPT> tag will always run before the Window_OnLoad event procedure.

Adding Code in a <SCRIPT> Section

The code in a <SCRIPT> section that is outside of a procedure will run automatically when the browser loads the Web page.

The following example code sets the value of the variable **X**, and then displays a message box when the page loads:

```
<SCRIPT LANGUAGE=VBScript>
    Dim X
    X = False
    MsgBox "Welcome!"
</SCRIPT>
```

Note The location of initialization code in the <SCRIPT> section determines when it will be run. When the page loads, the browser reads the code from the top of the page. This is particularly important when writing code that initializes object properties. For the object to be recognized, the initializing code must appear after the tag that defines the object.

Using the Onload Event

When an HTML page loads, the OnLoad event of the **Window** object will run. To use this event, you can either create a **Sub** procedure named Window_OnLoad or add the OnLoad attribute to the <BODY> tag.

To see a demonstration of how to use the Window_OnLoad event procedure, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

In the following example code, the Window_OnLoad event procedure initializes a form by calling the Initialize_ShipForm procedure:

```
Sub Window_OnLoad()
    Initialize_ShipForm
End Sub
Sub Initalize_ShipForm ()

End Sub
```

The following example code sets the Onload attribute of the <BODY> tag to call the Initialize_ShipForm procedure when the page loads:

```
<BODY LANGUAGE=JavaScript Onload = Initialize_ShipForm()>
```

For information about the **Window** object, see Window Object.

There are advantages and disadvantages of using ActiveX controls and Java applets on Web pages.

ActiveX Controls

When using ActiveX controls, you may want to consider these issues:

® Performance

ActiveX controls are compiled into native code, so they run faster than Java applets.

® Developer familiarity

Many Visual Basic and Visual C++ programmers already use ActiveX controls.

® Availability

There are many ActiveX controls that ship with Visual Basic or that can be purchased from a control developer.

® Browser support

ActiveX controls will run with any browser that supports the <OBJECT> tag. For browsers that support only the <EMBED> tag, a Netscape plug-in is available.

For information about browsers that do not support the <OBJECT> tag, see the technical article, The <EMBED> Tag in the Articles and White Papers section of the Library.

® Platform specific

ActiveX controls are compiled into native code, so they will work only on the platforms for which they were built. However, you can provide a file that downloads the correct control for the platform on which the browser is running.

For information about using ActiveX controls on multiple platforms, see the technical article, Packaging ActiveX Controls in the Articles and White Papers section of the Library.

For information about using ActiveX controls on the Apple Macintosh, go to the MacOLE Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

For a Microsoft press release about using ActiveX controls on a UNIX system, go to the Microsoft Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

® Security

ActiveX controls can be programmed to be malicious. To add security to ActiveX controls, you can sign the controls by obtaining a digital certificate from a Certificate Authority. A digital certificate provides accountability and authenticity. Microsoft Internet Explorer users can view the digital certificate of a control before deciding whether or not to download it.

For more information about securing against malicious ActiveX controls, see Using Signed Controls.

Java Applets

When using a Java applet, you may want to consider these issues:

® Reduced size

Java applets tend to be smaller than other types of executables, because the library functions are resident to the Java Virtual Machine (JVM) and are not brought in as overhead when the applet loads.

® Browser independent

The only requirement for a Java applet to run in a browser is that the browser should support the JVM. When an applet runs, it will appear to be run by the resident browser. However, the applet is actually run entirely by the JVM. All Java applets run in this way, regardless of the browser or the client computer that is being used.

® Platform independent

The Java language and Java Virtual Machine architecture were designed to run on any platform that has a Java-enabled browser and the Java Virtual Machine.

® Security

Java applets must abide by a strict security model. In this model, the applet runs in its own application space,

or “sandbox.” The sandbox model does not permit an applet to do the following tasks:

- Perform database or file I/O.
- Make calls to the operating system.
- Start or communicate with other applications outside of the current Web page.
- Open a communication channel to a server other than the server from which it was downloaded.

Because of these restrictions, it is extremely difficult for a Java applet developer to intentionally or accidentally create a malicious or dangerous applet.

Note Microsoft Internet Explorer categorizes Java classes as either trusted or untrusted. Untrusted classes run in their own application space, within the sandbox, so they cannot use COM services. Trusted classes are the only classes that can use COM services and are free of the sandbox constraints.

For information about how your Java applet can access COM services and features, go to the Microsoft SDK for Java Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

- Non-conformance of browsers

  Java is a rapidly changing language. Therefore, different Java-enabled browsers and JVMs may conform to different specifications. As Java matures, this problem should be reduced.

- Performance

  Java applets run slower than executables that are compiled into native code. Due to the interpreted nature of the language, this issue is especially noticeable in large, enterprise Java applets and applications. Recent technologies, such as just-in-time (JIT) compilers, partially address this issue.

Users of Microsoft Internet Explorer can control the types of active content they want to download on the Web pages they visit. They can disable all active content, or specify that only certain categories of content be allowed.

To specify certain categories of active content, users can click **Options** on the Microsoft Internet Explorer **View** menu, and then click the **Security** tab. Under **Active content**, users can specify which types of content they want to download and run on their computers, as shown in the following illustration:

{ewc MVIMG, MVIMAGE,!W04G100.bmp}

For example, if a user clears the option **Allow downloading of active content**, and selects the option **Enable ActiveX controls and plug-ins**, the user will only be able to use controls that are already installed on their computer.

Web pages that contain active content can cause security problems. To set the level of security you want Microsoft Internet Explorer to enforce on the Web pages you visit, click the **Safety Level** button under **Active content**. In the **Safety Level** dialog box, you can select from three levels of security: **High** (the default), **Medium**, or **None**.

To see an illustration of the **Safety Level** dialog box, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

The initial properties values you set for an ActiveX control define how the control will be displayed when the HTML page is first loaded by the browser.

To set initial object properties for an ActiveX control, you use <PARAM> tags.

The <PARAM> tag uses the following syntax:

```
<PARAM NAME="ParameterName" VALUE="Value">
```

For each property, you use a separate <PARAM> tag.

The following example code shows the <OBJECT> tag for the ActiveX **Label** control. The control has initial property values set for Angle, Alignment, and Caption.

```
<OBJECT
    classid="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
    id=lblActiveLbl
    width=250
    height=250
>
<PARAM NAME="Angle" VALUE="90">
<PARAM NAME="Alignment" VALUE="2">
<PARAM NAME="Caption" VALUE="Hello, World!">
</OBJECT>
```

Note To determine the property names and values supported by an object, see the Help file for the object, or use the Object Browser in Microsoft Visual Basic 4.0 or later.

The CODEBASE attribute for the <OBJECT> tag contains an absolute or relative URL that points to the .ocx file for a given control.

If an ActiveX control does not rely on any files other than the .ocx file, you can specify the location of the .ocx file with the CODEBASE attribute.

The following example code downloads an ActiveX control by referencing the .ocx file:

```
<OBJECT
    CLASSID="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
    CODEBASE="http://server/control.ocx">
</OBJECT>
```

If an ActiveX control relies on additional DLLs to run, you must supply a .cab file that contains all necessary files, and an .inf file that specifies where to install each of the files. For information about how to use .cab files, see the technical article, Packaging ActiveX Controls in the Articles and White Papers section of the Library.

The following example code downloads and installs an ActiveX control by referencing a .cab file:

```
<OBJECT
    CLASSID="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2
    CODEBASE="http://server/control.cab">
</OBJECT>
```

Specifying a Version Number

The CODEBASE attribute also enables you to include a version number to make sure that users download only a specific version of a control.

In the following example code, the CODEBASE attribute specifies a version number of 4.70.1165 in the pop-up menu control.

```
<OBJECT
    CLASSID="CLSID:7823A620-9DD9-11CF-A662-00AA00C066D2"
    CODEBASE="iemenu.ocx#Version=4,70,0,1165">
```

The format of the version number is N,N,N,N. If you do not specify a version number, Internet Component Download will use any version of the control that is installed on the user's computer.

To determine the version number of a control, right-click the .ocx file in Windows Explorer, and then click **Properties**. In the **Properties** dialog box, click the **Version** tab, and then locate the version number. If the version number is not in the format N,N,N,N, you can run the ActiveX Version Utility included with this CD-ROM.

For information about using this utility, see ActiveX Version Utility in the Library included on this CD-ROM.

To make sure that a control will always download onto the user's computer, specify the version number -1,-1,-1,-1.

In addition to using the Visual InterDev Source Editor or the FrontPage Editor to add ActiveX controls to a Web page, you can also add the <OBJECT> tag manually to a Web page.

Note Some browsers do not support the <OBJECT> tag. To insert ActiveX controls on Web pages for browsers that do not support the <OBJECT> tag, use the <EMBED> tag. For information about how to use the <EMBED> tag, see The <EMBED> Tag article in the Library section.

<OBJECT> Tag Syntax

The <OBJECT> tag has attributes that define the ActiveX control, the location of an object, and how the object will be displayed on an HTML page. These attributes are listed in the following paragraphs.

CLASSID

When an ActiveX control is installed on a user's computer, it is registered in the system registry with a unique class ID. The CLASSID attribute, which contains the class ID, is the only required attribute for the <OBJECT> tag.

The syntax for the CLASSID attribute is:

```
CLASSID = "clsid:12345678-1234-1234-1234-123456789012"
```

The following example code shows the <OBJECT> tag for the **Calendar** control:

```
<OBJECT
    classid="clsid:8E27C92B-1264-101C-8A2F-040224009C02"
>
</OBJECT>
```

Visual InterDev and FrontPage automatically search the registry for the class ID and write it in the generated <OBJECT> tag. You can determine the class ID for a control manually by using one of the following tools:

® Registry Editor (RegEdit)

This tool enables you to change settings in your system registry. You can also use it to view and copy the class ID for an object.

® OLE Viewer (Ole2View)

This tool is installed with the Win32 Software Development Kit (SDK). You can use the **Copy HTML <object> Tag to Clipboard** command on the **Object** menu to copy the <OBJECT> tag to the Clipboard, and then paste the tag into an HTML page or an Active Server Page.

To download the OLE Viewer, go to the Microsoft Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

ID

The ID attribute enables you to refer to the object from the Visual Basic Scripting Edition (VBScript) language.

The following example code shows how to set the ID attribute of the **Calendar** control:

```
<OBJECT
    classid="clsid:8E27C92B-1264-101C-8A2F-040224009C02"
    id=cldCalendar2
>
</OBJECT>
```

CODEBASE

The CODEBASE attribute is a URL that points to a file containing the implementation of an object. For information about this attribute, see Setting the CODEBASE Attribute.

WIDTH, HEIGHT, ALIGN, HSPACE, VSPACE, and BORDER

These attributes affect the way in which an object is positioned and sized on an HTML page, and whether or not it is contained in a border.

Supplying an Alternative for the <OBJECT> Tag

For browsers that do not support the <OBJECT> tag, or for users who have not enabled ActiveX controls in their Web browser, you should supply an alternative way to provide the same functionality. If a Web browser does not display ActiveX controls, it will display any HTML tags you place between the <OBJECT> and </OBJECT> tags.

Note If a browser supports the <OBJECT> tag, but the user does not install the ActiveX control, the browser will display an invalid control icon, and not the alternative functionality.

For example, for browsers that do not display the **Calendar** control, you can place a text box control on the Web page for a user to enter the date, as shown in the following code:

```
<OBJECT
    classid="clsid:8E27C92B-1264-101C-8A2F-040224009C02"
>Best if viewed with Internet Explorer 3.0.<P>
Type a date here:
<INPUT TYPE=TEXT NAME="Date" MAXLENGTH=10 SIZE=10>
</OBJECT>
```

Some browsers do not support the <OBJECT> tag, and cannot view ActiveX controls. To insert ActiveX controls into Web pages for browsers that do not support the <OBJECT> tag you can use the <EMBED> tag. The ScriptActive plug-in, developed by the NCompass Corporation, enables you to insert ActiveX controls with the <EMBED> tag.

To see a demonstration of how to use the ScriptActive plug-in to view and use an ActiveX control, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

If the ScriptActive plug-in is installed on a user's computer, you can add ActiveX controls to a Web page to work either with the <OBJECT> tag, or with the <EMBED> tag.

For example, the following code uses the ScriptActive plug-in to view the **Calendar** control.

```
<EMBED TYPE="application/x-oleobject"
    NAME="Calendar"
    classid="clsid:8E27C92B-1264-101C-8A2F-040224009C02"
    CODEBASE="http://server/controls/mscal80.ocx"
    WIDTH=300
    HEIGHT=300
    PARAM_FirstDay=2
    PARAM_ShowTitle=0
>
```

For information about the ScriptActive plug-in, jump to the NCompass Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

You can add an ActiveX control to a Web page that can be viewed by browsers that support either the <OBJECT> tag or the <EMBED> tag. If a browser supports the <OBJECT> tag, it will ignore the <EMBED> tag. If a browser does not support the <OBJECT> tag, it will ignore the <OBJECT> tag and the associated <PARAM> tags.

The following HTML example code shows how different browsers interpret the<OBJECT>and <EMBED> tags.

{ewc MVIMG, MVIMAGE,!W04g070.bmp}

To insert a Java applet, you can add <APPLET> tags in an HTML file. The <APPLET> tag has a number of attributes that define the applet, its location, and how it will be presented on the HTML page.

The following attributes are required:

® CODE

Identifies the name of the .class file to be downloaded.

Note The CODE attribute is case sensitive.

® WIDTH

Specifies the width of the area in which the applet will run.

® HEIGHT

Specifies the height of the area in which the applet will run.

The following example code shows how an <APPLET> tag adds the Java Outline applet to a Web page:

```
<APPLET
    CODE=Outline.class
    HEIGHT=150
    WIDTH=200>
</APPLET>
```

This example code shows the syntax for setting attributes in the <APPLET> tag:

```
<APPLET
    ALIGN=LEFT|CENTER|RIGHT|TOP|MIDDLE|BOTTOM
    CODE=appletFile
    CODEBASE=codebaseURL
    HEIGHT=pixels
    HSPACE=pixels
    NAME=appletInstanceName
    VSPACE=pixels
    WIDTH=pixels>
<PARAM NAME=AttributeName VALUE=value...>
alternate HTML
</APPLET>
```

The following table defines these attributes.

Attribute	Definition	Required
ALIGN	Specifies the alignment of the applet in relation to the text. Possible values are LEFT, CENTER, RIGHT, TOP, MIDDLE, and BOTTOM. The default value is LEFT.	No
ARCHIVE	Name of the .zip file that contains an archive of .class files used by the applet.	No
CODE	Name of the file that contains the compiled applet subclass. This file is relative to the base URL of the applet (the folder in which the applet is located). The file cannot be an absolute value.	Yes
CODEBASE	Specifies the base URL of the applet. If this attribute is not specified, the URL of the HTML page will be used.	No
WIDTH and HEIGHT	Specifies the initial width and height (in pixels) of the applet display area. This area does not	Yes

	include any windows or dialog boxes invoked by the applet.	
VSPACE and HSPACE	VSPACE specifies the number of pixels above and below the applet. HSPACE specifies the number of pixels on each side of the applet.	No
NAME	Identifies an applet to other applets on the same HTML page.	No

Supplying an Alternative for the <APPLET> Tag

For browsers that do not support the <APPLET> tag, or for users who have not enabled Java applets in their Web browser, you should supply an alternative way to provide the same functionality. If a Web browser does not display Java applets, it will display any HTML tags you place between the <APPLET> and </APPLET> tags.

In the following example code, the alternative HTML informs users that the Java applet is missing, and uses hyperlinks to provide the same functionality as the Java Outline applet:

```
<APPLET
    CODE=Outline.class
    HEIGHT=150
    WIDTH=200>
You're missing the Java Outline applet.
<UL>
<LI><A HREF="default.htm">State University Home Page</A></LI>
<LI><A HREF="majors.htm">State University Majors</A></LI>
</UL>
</APPLET>
```

In Visual InterDev, you can add a Java applet by using the FrontPage Editor or by adding the <APPLET> tag with the Visual InterDev Source Editor.

In this section, you will learn how to add a Java applet to a Web page by using the FrontPage Wizard and <APPLET> tag.

This section includes the following topics:

With the Front Page Editor, you can easily add a Java applet to an HTML page.

To add a Java applet with the FrontPage Editor

1. On the **Insert** menu, click **Other Components**, and then click **Java Applet**.

   To see an illustration of the FrontPage **Java Applet Properties** dialog box, click this icon.
   {ewc mvimg, mvimage,!illust.bmp}
2. In the dialog box, enter the name of the applet source file that you want to insert.

   Java applet source files usually have a .class extension.
3. If necessary, enter a location for downloading a .cab file.
4. Under **Applet Parameters**, set initial properties for a control.

   Java does not provide a way for you to view valid properties and values for a control. To display valid property names and legal values for each property, refer to the online documentation included with the Java applet.
5. Enter size and alignment property values, and then click OK.

   To save the layout of the page, FrontPage inserts an actual-size placeholder icon.

To edit a Java applet

1. Right-click the control you want to edit, and then click **Java Applet Properties**.

   This displays the **Java Applet Properties** dialog box for the selected object.
2. Edit the Java applet properties, and then click OK.

   FrontPage modifies the <APPLET> tag for the applet to reflect your changes.

To see a demonstration of how to add a Java applet to an HTML page by using the FrontPage Editor, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Developers license some controls to protect them from malicious tampering. When you use licensed controls in a Visual Basic application, the application supplies a run-time license automatically. When you use these licensed controls on an HTML page, you must invoke the License Control Manager to supply the control with the license it needs to function.

Note A run-time license is different from a design-time license. A run-time license for a control is used as part of an HTML solution. A design-time license is received when you download and run a Setup program for an application. You must have design-time licenses to create HTML pages, and users must have run-time licenses.

The Licensed Control Manager

Each HTML page that uses a licensed control requires a license (.lpk) file. The Licensed Control Manager, which is included with Microsoft Internet Explorer 3.0, will use only the first .lpk file on a page. Therefore, the .lpk file must contain run-time licenses for all licensed controls on the HTML page.

You can use the License Package Authoring Tool (Lpk_Tool.exe) to create .lpk files. The License Package Authoring Tool is included in the Microsoft ActiveX SDK, Microsoft Visual Basic 5.0, and the Library of this CD-ROM. To install this tool, see License Package Authoring Tool in the Resources\Tools section of the Library.

u **To add a licensed control with the Licensed Control Manager**

1. Create an .lpk file by running Lpk_Tool.exe.
2. Place the .lpk file in the same folder as the Web page.
3. Add an <OBJECT> tag for the Licensed Control Manager to the Web page.
4. Add a <PARAM> tag and set the NAME attribute to LPKPath and the VALUE attribute to the name of the .lpk file.

The following example HTML shows the <OBJECT> and <PARAM> tags for the Licensed Control Manager and an .lpk file for a spin control:

```
<OBJECT CLASSID="clsid:5220CB21-C88D-11CF-B347-00AA00A28331">
    <PARAM NAME="LPKPath" VALUE="spin32.lpk">
</OBJECT>
```

The <OBJECT> tag invokes the Microsoft Licensed Control Manager, which reads the .lpk file and verifies the run-time license.

To see a demonstration of how to use Lpk_Tool.exe and add the <OBJECT> tag for the Licensed Control Manager, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Note The run-time license is verified only when the control is loaded. Refresh will not necessarily check the license file.

ActiveX controls (formerly known as OLE controls) are component objects that can be inserted on Web pages or other ActiveX container applications. These controls can be built with multiple languages, such as C, Visual C++, Visual Basic, and Java. ActiveX controls are typically built as dynamic link libraries (DLL), and are compiled with an .ocx extension.

The following illustration shows how ActiveX controls can be inserted on a Web page.

{ewc MVIMG, MVIMAGE,!W04G020.bmp}

ActiveX Controls in an Application and on a Web Page

To run an ActiveX control on a Web page or in an application, the control must already be installed and registered on the user's computer. If the control requires additional functionality from DLLs, the DLLs must also be installed on the user's computer.

When you create a stand-alone application that uses ActiveX controls, you typically create a Setup program. This program will install the application, the .ocx files, and any necessary DLLs on the user's computer. The Setup program will also register the ActiveX controls, so when the application runs, the .ocx files will be ready to use.

Web pages do not have Setup programs, so the .ocx files and DLLs must already be installed on the user's computer before the Web page can run properly.

To insert controls on a Web page and ensure that they are properly installed, you can use the HTML <OBJECT> tag and set the CODEBASE attribute. For information about the <OBJECT> tag and the CODEBASE attribute, see The <OBJECT> Tag) and Setting the CODEBASE Attribute.

When Microsoft Internet Explorer encounters an <OBJECT> tag in HTML code, it searches for the control on the user's computer. If the control has not been installed, Microsoft Internet Explorer downloads the control from the location specified with the CODEBASE attribute, and then installs the control.

Types of ActiveX Controls

You can add any ActiveX control to an HTML page. Some controls, however, are optimized for use on the Internet. These controls have less overhead, so they are more effective in lower bandwidth systems (such as the Internet), and are more quickly transferred to the client.

Other ActiveX controls have been created specifically for developing Internet-aware applications, and are integrated with Internet services and protocols. You can use these controls to access Internet services directly, rather than from an HTML page. For example, the HTTP, FTP, and Gopher protocols have been implemented as ActiveX controls.

For a list of available ActiveX controls, go to the Microsoft ActiveX Component Gallery Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

By default, Java applets are downloaded one .class file at a time and in an uncompressed format. As applets become more complex and rely on more than one file, this downloading method becomes slow and inefficient.

Using CAB Files

One way of reducing download time is to use a cabinet (.cab) file to compress and distribute multiple .class files.

To use a .cab file for a Java applet, you add a <PARAM> tag with the name cabbase, and a value that identifies the name of the .cab file, as shown in the following example code:

```
<APPLET
    CODE=Outline.class
    WIDTH=150
    HEIGHT=200>
    <PARAM NAME=cabbase VALUE=outline.cab>
</APPLET>
```

This code informs the Java Virtual Machine that .class files of the Java applet should be extracted from the .cab file, instead of being loaded one .class file at a time. A .cab file must be located in the same folder as the HTML page or in the location specified with the CODEBASE attribute.

Using ZIP Files

Microsoft Internet Explorer is the only browser that supports .cab files. To make download time faster in other browsers, you can create (but not compress) a .zip file of all the .class files needed for your Java applet. The Java Virtual Machine reads the .zip file and extracts the .class files in it.

To use a .zip file, set the ARCHIVE attribute of the <APPLET> tag to the name of the .zip file.

You can set the ARCHIVE attribute and add the cabbase <PARAM> tag, as shown in the following example code:

```
<APPLET
    CODE=Outline.class
    WIDTH=150
    HEIGHT=200
    ARCHIVE=outline.zip>
    <PARAM NAME=cabbase VALUE=outline.cab>
</APPLET>
```

In this section, you will learn how users can set options in Microsoft Internet Explorer to protect themselves from suspect controls.

The ActiveX controls on a Web page can cause problems for users in three ways.

First, ActiveX controls are downloaded from Web servers, so the source code can be altered before it is downloaded.

Second, ActiveX controls run as native code on a user's computer, so there is no limit to what the control can modify on the computer.

Third, script code on a Web page can manipulate the properties and methods of a control, which can potentially cause damage to a user's computer.

This section includes the following topics:

Even if a control has been signed, it does not guarantee that the control is safe to be used on a Web page. There are two other ways in which you can guarantee the safety of a control. You can make sure that the control is safe to initialize, and also that it is safe to be used in the script on the Web page.

Safe for Initializing

Controls that can be initialized with values in HTML <PARAM> tags must be marked as safe for initializing before Microsoft Internet Explorer will read the <PARAM> tags.

When Microsoft Internet Explorer encounters a control on a Web page that has initialization tags, it queries the control to determine if it is safe for initialization. The result of this query, combined with the safety level set by the user, determines whether the control is initialized or not.

The following table shows whether a control will be initialized or not, depending on which safety level the user has set. To see the dialog box displayed in each case, click the associated icons in the table.

Level	Safe for initializing	Not safe for initializing
High	Control initializes.	User is warned of unsafe content in the **Potential safety violation avoided** dialog box. The control will display and function, but is not initialized and will use default values for its persistent properties. The control cannot be scripted. To see the **Potential safety violation avoided** dialog box, click this icon. {ewc mvimg, mvimage,!illust.bmp}
Medium	Control initializes.	User is warned of unsafe content with the **Potential safety violation** dialog box and can decide whether or not the control should be initialized. To see the **Potential safety violation** dialog box, click this icon {ewc mvimg, mvimage,!illust.bmp}
None	Control initializes.	Control initializes.

Safe for Scripting

When you mark a control as safe for scripting, you guarantee that the object model of the control will not cause a safety problem, either in the form of data corruption or as security leaks.

Marking a control means that there will not be any properties or methods available for use by the script code. For example, a control that allows information to written to the hard disk should not be marked as safe for scripting.

When Microsoft Internet Explorer encounters a control on a Web page, it queries the control to determine if it is safe for scripting. The result of this query, combined with the safety level set by the user, determines whether the control is available to be scripted.

The following table shows whether a script will be run or not, depending on which safety level the user has set. To see the dialog box displayed in each case, click the associated icons in the table.

Level	Safe for scripting	Not safe for scripting
High	Scripts run.	If any script refers to the control, the script fails and the user is warned with the **Internet Explorer Script Error** dialog box. To see the **Internet Explorer Script Error** dialog box, click this icon. {ewc mvimg, mvimage,!illust.bmp}
Medium	Scripts run.	User is warned of unsafe content with the **Potential safety violation** dialog box and can

		decide whether or not to allow scripts to run. To see the **Potential safety violation** dialog, click this icon. {ewc mvimg, mvimage,!illust.bmp}
None	Scripts run.	Scripts run.

To ensure a safe and reliable method of distributing software over the Internet, Microsoft Corporation and other software companies have developed a method of digitally signing code.

Signing a Control

To sign the code for a control, you obtain a digital certificate from a Certificate Authority. A control that is signed with a digital certificate is not necessarily a safe control. The digital certificate only guarantees that the code being downloaded has been built and signed by a qualified developer, and that it has not been tampered with or corrupted.

For more information about digital certificates, see Using Certificates in Chapter 10: Controlling Access to a Web Site.

Setting the Safety Level

Users can determine whether or not they want to download and run controls by setting the safety level and finding out if the control has been signed with a digital certificate.

The following table shows whether a control will be downloaded, depending on which safety level the user has set, and whether or not the control has been signed. To see the dialog box that appears in each case, click the associated icons in the table.

Level	Signed control	Unsigned control
High	User is shown an **Authenticode** dialog box with certificate information, and can decide whether or not to download the control. To see the **Authenticode** dialog box, click this icon. {ewc mvimg, mvimage,!illust.bmp}	User is shown the **Potential safety violation avoided** dialog box, which warns of unsafe content, and the control is not downloaded. To see the **Potential safety violation avoided** dialog box, click this icon. {ewc mvimg, mvimage,!illust.bmp}
Medium	User is shown an **Authenticode** dialog box with certificate information, and can decide whether or not to download the control. To see the **Authenticode** dialog box, click this icon. {ewc mvimg, mvimage,!illust.bmp}	User is shown the **Authenticode Security Technology** dialog box, which warns of unsafe content. The user can decide whether or not to download the control. To see the **Authenticode Security Technology** dialog box, click this icon. {ewc mvimg, mvimage,!illust.bmp}
None	User does not receive any notification, and the control is downloaded.	User does not receive any notification, and the control is downloaded.

If a control is not downloaded to the user's computer, the following placeholder image will replace the control: {ewc MVIMG, MVIMAGE,!W04g065.bmp}

For more information about Authenticode technologies, go to the Microsoft Internet Security Framework Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

The Java language, a derivative of C++, was developed by Sun Microsystems, Inc. to be a robust, secure, and cross-platform language for use on the Internet. Java can be used to create stand-alone applications and applets.

Applets are a type of object that you can use to create an interactive HTML page. Java applets are small, reusable programs that expose an interface and are run in a certain type of container, such as a Web browser.

How Java Applets Work on Web Pages

When you compile Java source (.java) files, the Java compiler creates Java bytecode (.class) files. When these .class files are downloaded to the user's computer, they are interpreted locally by the Java Virtual Machine (JVM).

You can install the JVM as part of a Web browser, or download it from the Internet.

To add Java applets on an HTML page, you use the <APPLET> tag. The CODE parameter of this tag specifies the .class file of the applet. To run an applet in a Web browser, the user's computer must have the JVM installed, and the browser's security must be set to enable the applets to run.

The following illustration shows how Java files are interpreted.

{ewc MVIMG, MVIMAGE,!W04G050.bmp}

For a list of some of the available Java applets, go to the Gamelan Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

The *Mastering Microsoft Visual J++* CD-ROM teaches you how to create Java applets. For information about this title, see Mastering Microsoft Visual J++ in the Resources section of the Library.

To complete the lab exercises in this chapter, you must have the required software. For detailed information about the labs and setup for the labs, see Labs in this course.

For background information on this lab, click each of these topics:

Objectives

Prerequisites

Lab Setup

To see a demonstration of the lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

To see a diagram of how the files you edit in this lab will fit into the State University Web site, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Estimated time to complete this lab: **60 minutes**

Note There are project and solution files associated with this lab. If you installed the labs during Setup, these files are in the folder <*Install Folder*>\Labs\Lab04 on your hard disk. If you did not install the labs during Setup, you can find them in the \Labs\Lab04 folder of the *Mastering Web Site Development* CD-ROM.

Exercises

The following exercises provide practice working with the concepts and techniques covered in Chapter 4: Using Objects on Web Pages.

Exercise 1: Adding ActiveX Controls

In this exercise, you will add image controls and a pop-up menu control to the State University home page.

Exercise 2: Adding a Java Applet

In this exercise, you will create a new page that displays State University's animated mascot.

Exercise 3: Using a Licensed Control

In this exercise, you will add an .lpk file for the ActiveX controls in the file mascot.htm to the StateU project, and then add the <OBJECT> tag for the License Manager to read the .lpk file.

After completing this lab, you will be able to:

® Add an ActiveX control to a Web page.
® Set design-time properties of ActiveX controls.
® Add the tag for the License Manager to a Web page.
® Add a Java applet to a Web page.

Before working on this lab, you should be familiar with the following concepts:

® Adding files to a Visual InterDev Web project.

® Editing files with the Visual InterDev Source Editor and the FrontPage Editor.

To complete this lab, you need the following:

- The Visual InterDev StateU Web project.
- These files in the StateU project: home.htm and mascot.htm.
- These image (.gif) files in the images folder of the StateU project: bullet, history, math, music, sulogo, sutitle, mascot, mascotlogo, and img0001.gif–img0018.gif.

There are three majors offered at State University: Math, History, and Music. Students who use this Web site need an easy way to find out which classes are offered for each major.

In this exercise, you will add **Image** controls for each of the three majors, and a pop-up menu control that lists the classes for each major. You will then add script code to display the pop-up menu when a user selects a major that is represented by an image.

To see an example of how your finished Web page should look, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Add files to the Web project

1. In Visual InterDev, open the StateU project.
2. Copy the two controls isctrls.ocx and iemenu.ocx from the folder \MWD\Labs\Lab04 to the controls folder in the StateU project.

Add Image controls

Add the ActiveX **Image** control for the Math major, and then add **Image** controls for the History and Music majors.

1. In Visual InterDev, open home.htm with the FrontPage Editor.
2. Insert an ActiveX **Image** control for the Math major at the marked location in the file.
   a. On the **Insert** menu, click **Other Components**, and then click **ActiveX Control**.
   b. In the **Pick a Control** drop-down list box, select **Microsoft ActiveX Image Control**.
   c. Set **Name** to imgMath.
      This adds the ID attribute to the <OBJECT> tag.
   d. Set **Code Source** to controls/isctrls.ocx.
      This adds the CODEBASE attribute to the <OBJECT> tag.
   e. To display the ActiveX Control Editor and the **Properties** dialog box for the control, click the **Properties** button, and then set values for the properties listed in the following table.

Property	Value
AutoSize	True
PicturePath	images/math.gif

Note FrontPage displays an **Invalid Property Value** dialog box when you set the **PicturePath** property of the **Image** control. Despite the error message, the .gif file will display in the **Image** control.

---

3. Insert ActiveX **Image** controls for the History and Music majors.
   a. Set the **Name** of the controls to imgHistory and imgMusic, respectively.
   b. Set **Code Source** to controls/isctrls.ocx.
   c. Set the **PicturePath** properties of the controls to images/history.gif and images/music.gif, respectively.
4. Save your changes to home.htm.
5. Close the FrontPage Editor, and then preview home.htm in the Visual InterDev InfoViewer.

For more information about inserting ActiveX controls on a Web page, see Using ActiveX Controls.

Add a pop-up menu control

In the file home.htm, add a pop-up menu control that will list the classes available for each major.

1. In Visual InterDev, open home.htm in the Visual InterDev Source Editor.
2. Right-click the file after the last </OBJECT> tag, and then click **Insert ActiveX Control**.
3. In the **Insert ActiveX Control** dialog box, select the **PopupMenu** object, and then click OK.

Note If the **PopupMenu** object does not appear in the list of registered controls, you will need to register the iemenu.ocx control on your development computer. To register the control, in the Windows Explorer, double-click the control in the folder C:\Program Files\DevStudio\MyProjects\StateU\controls.

Or, you can download the **Microsoft Popup Menu** control from the ActiveX Component Gallery Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

a. Set the CODEBASE attribute to the controls/iemenu.ocx.

b. Set the ID attribute to popClasses.

c. Close the **Properties** dialog box and the Visual InterDev Object Editor.

4. To hide the pop-up menu control until you want to display it:

   a. Delete the WIDTH and HEIGHT attributes of the <OBJECT> tag.

   b. Delete the <PARAM> tags for the properties _ExtentX and _ExtentY.

   To see an illustration of how the code for the <OBJECT> tag should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

5. To add menu items to the pop-up menu, add <PARAM> tags, as shown in the following example code:

```
<PARAM NAME="MenuItem[0]" VALUE="Class 1">
<PARAM NAME="MenuItem[1]" VALUE="Class 2">
<PARAM NAME="MenuItem[2]" VALUE="Class 3">
<PARAM NAME="MenuItem[3]" VALUE="Class 4">
```

Note In the lab for Chapter 5: Adding Client-Side Script, you will write client-side script to add classes dynamically to the pop-up menu for each selected major.

   For more information about adding <PARAM> tags, see Setting Design-Time Properties.

6. Save your changes to home.htm.

u Add script to display the pop-up menu

Add script code to display pop-up menus when a user selects one of the images.

1. With the Visual InterDev Source Editor, start the Script Wizard by right-clicking anywhere in the home.htm file, and then clicking **Script Wizard**.

   To see an illustration of the **Script Wizard** dialog box, click this icon.
   {ewc mvimg, mvimage,!illust.bmp}

   For information about using the Script Wizard, see Using the Script Wizard in Chapter 5: Adding Client-Side Script.

2. At the bottom of the dialog box, click the **Code View** option.
3. Expand the list of events for the **imgMath** control by clicking the plus sign (**+**).
4. Select the Click event of the **imgMath** control.

   A template for the imgMath_Click event procedure will appear in the Script Wizard.

5. In the imgMath_Click event procedure, display the pop-up menu by adding the following script:

```
popClasses.PopUp
```

   To see an illustration of how the full syntax of the event procedure should look in the home.htm file, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

6. To display the pop-up menu when a user clicks the **imgHistory** and **imgMusic** controls, repeat steps 3 through 5.
7. To close the Script Wizard, click OK.

   Visual InterDev writes the Visual Basic Scripting Edition (VBScript) code in the file home.htm.

8. Save your changes to home.htm.

u Test the Web page

1. Preview home.htm in the Visual InterDev InfoViewer.
2. Click the images for the different majors.

To create a Web application that can access a database, you first create a data source. The data source specifies the database server and the database you want to access.

After you have created a data source, you add a data connection to your Web project. The data connection specifies which data source will be used for your Web application.

When you add a data connection to a Web project, Visual InterDev generates script in the Global.asa file. The script saves information about the data connection in variables of the **Session** object. This information can then be used in ASP pages when creating a database connection.

In this section, you will learn how to create a data source, and how to add a data connection to your Visual InterDev project. You will also review the script that Visual InterDev adds to the Global.asa file when you add a data connection to a project.

This section includes the following topics:

When you create a Web application that accesses a database, you often need to work with the database while creating the application. For example, you may want to determine how tables are related, verify a field name, or run a query.

When you add a data connection to your Web project, Visual InterDev adds a **Data View** tab in the Workspace window. DataView displays the objects contained in a database, such as tables, views, and stored procedures, and provides a graphical environment for working with the database. In DataView, you can view and modify the records in a table, add new tables, change the structure of a table, and create and execute stored procedures.

To see an illustration of the **Data View** tab, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

In this section, you will learn how to use Data View to view and modify the data in a database, change the structure of a table, add a new table, and create a database diagram that shows how the tables in a database are related.

This section includes the following topics:

Visual InterDev provides the Data Form Wizard, which creates Active Server Pages (.asp files) and generates the necessary ActiveX Data Objects (ADO) code to access and update records in a database.

You can use the Data Form Wizard to create simple database entry forms for your Web application without having to write any code. For example, you can create forms that a database administrator can use for basic table maintenance.

The Data Form Wizard also provides an excellent tool for learning how to write ADO code. You can run the wizard, and then review the generated ADO code.

If you want greater control over the format of your Web page and how the data is displayed, you can use the design-time ActiveX data range controls. For information about data range controls, see Using the Data Range Controls.

To see an illustration of a form that has been created with the Data Form Wizard, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

In this section, you will learn how to run the Data Form Wizard to create a Web page that users can use to view and update the records in a database.

This section includes the following topics:

To access a database from your Web application, you must first add a data connection to your Web project in Visual InterDev. Each data connection identifies a single data source that can be accessed by your project.

To see a demonstration of how to add a data connection to a project, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

To add a data connection to a Web project

1. In the Visual InterDev project workspace, click the **FileView** tab.
2. Right-click either the Global.asa file or the root folder of your project.
3. Click **Add Data Connection**.
4. In the **Select Data Source** dialog box, click the **Machine Data Source** tab.
5. Select a data source name, or click **New** to create a new data source.
   {ewc mvimg, mvimage,!tip.bmp}
6. If the data source requires a login, enter a valid database login and password.

   This login will be used by Visual InterDev when you work with the database at design time.
7. On the **General** tab of the **Properties** dialog box, enter a name for the data connection.

   This name will be listed when you switch to FileView.
8. To select a different data source name, modify the connection string or click **Connection Builder**.
9. On the **Run-time** tab, enter the user name and password that will be used by the .asp files to access the database.

   This login will be used at run time when a user requests the .asp file, or when you preview the .asp file in Visual InterDev.
10. Close the **Data Connection Properties** dialog box.

    A data connection will be added under the Global.asa file in FileView. To change properties of the data connection, double-click the **Data Connection** icon.

    A **Data View** tab is also added to the Visual InterDev project workspace.
11. To display the tables and fields available in the data source, click the **Data View** tab, and then expand the folders and files by clicking the plus sign (**+**) next to the data source name.

Note If you are using an SQL Server or a Microsoft Access database on a different computer from the Web server, you must enable the NT Guest account on the computer where the database resides, or set up a special account for each user who will log in.

For information about this scenario, see the article #Q152828, IIS Queries to SQL Server Generate Error 1326 in the Knowledge Base section of the Library.

When you add a data connection to your Web project, Visual InterDev generates the Session_OnStart event procedure in the Global.asa file for the project.

This procedure runs on the Web server when the first page is requested by a new user. The connection information is stored in session properties, so you can create connections by using the session properties in your Active Server Page (.asp) files.

To view the script in the Global.asa file, switch to FileView, and then double-click the file. Your Global.asa script will look similar to the following one:

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">
Sub Session_OnStart
    '==Visual InterDev Generated - DataConnection startspan==
    '--Project Data Connection
        Session("StateU_ConnectionString") = "DSN=StateU;UID=sa;PWD=;APP=Microsoft
(R) Developer Studio;WSID=MARYAK;DATABASE=StateU"
        Session("StateU_ConnectionTimeout") = 15
        Session("StateU_CommandTimeout") = 30
        Session("StateU_RuntimeUserName") = "sa"
        Session("StateU_RuntimePassword") = ""
    '==Visual InterDev Generated - DataConnection endspan==
End Sub
</SCRIPT>
```

When you connect to a Microsoft SQL Server database from Visual InterDev, you can create a diagram of the database. A database diagram is a graphical representation of the tables in a database and the relationships between tables.

You can use a database diagram to create and modify table relationships, modify table properties, and create new tables.

To see an illustration of a database diagram, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

To see a demonstration of how to create a database diagram, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

To create a database diagram

1. In Data View, click **New Database Item** on the **Insert** menu.
2. Click **Database Diagram**, and then click OK.
3. Drag the tables you want to display into the diagram.

   If a relationship exists between any two tables, a relationship line will appear automatically.
4. On the **Database Diagram** toolbar, click the **Show Column Properties** button to display the properties of each column in a table.

   To modify table properties, enter new values in the table design grid. To join two tables so that they have a relationship to one another, drag a field from one table to the field of another table in the database diagram. To change the relationship between tables, right-click the join line.

As you modify the database diagram, Visual InterDev generates SQL script that will implement your changes. After you close the database diagram and save changes, Visual InterDev will run the SQL script and the changes will be made to the database.

If you want to view or save the SQL script generated by Visual InterDev, click the **Save Change Script** button on the **Database Diagram** toolbar.

To add or change data in a database, open the table by switching to Data View, and then double-clicking the table name.

To modify a record, enter a new value in the appropriate column. The record will be updated when you move focus to another record. To add a new record, enter data in the blank record at the end of the table.

To see a demonstration of how to use Data View to modify and insert data in a database, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

In Data View, you can view and modify the structure of existing items in a database and create new database items.

To see a demonstration of how to modify the structure of a database, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Changing a Table Design

To view or modify the structure of a table, switch to Data View, right-click the table name, and then click **Design**. A grid shows the design of the table. To modify the definition of each column, type in the table grid.

To see an illustration of an open table in design view, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

To modify the general properties of a table, such as relationships, indexes, and constraints, click the **Properties** button on the **Table** toolbar, and complete the information in the **Properties** dialog box.

To see an illustration of the **Properties** dialog box, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

As you modify the table design, Visual InterDev generates SQL script that will implement your changes. When you close the table and save changes, Visual InterDev runs the SQL script and makes the changes to the table.

If you want to view or save the SQL script, click the **Save Change Script** button on the **Table** toolbar.

Creating a New Table

You can use Visual InterDev to add new items, such as a table, stored procedure, or trigger to Microsoft SQL Server databases.

To add a new database item, click **New Database Item** on the **Insert** menu, and then select the type of object you want to create.

To see a demonstration of how to create a data form with the Data Form Wizard, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

To run the Data Form Wizard

1. In Visual InterDev, click **New** on the **File** menu.
2. On the **File Wizards** tab, select **Data Form Wizard**, and type a file name and location for the files that the wizard will create.

   The file name will be used as the prefix for each of the files created by the wizard.
3. Select the data connection that the wizard will use.
4. Depending on the records you want to retrieve, select **Table**, **View**, or **SQL Statement**.
   - To retrieve all records from a single table, select **Table**.
   - To retrieve all records from a predefined view, select **View**.
   - To enter an SQL statement to retrieve records, select **SQL Statement**.
     {ewc mvimg, mvimage,!tip.bmp}
5. Specify the edit and viewing options for your data form.
6. Select a theme for the way you want the pages to look, and then click **Finish**.

Depending on the options you selected in the wizard, up to three .asp files will be created with prefixes of the file name you specified. The files will be named *filename*Form.asp, *filename*List.asp, and *filename*Action.asp.

For information about how to use parameter queries with the Data Form Wizard, go to the Microsoft Visual InterDev Samples Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

The Data Form Wizard generates up to three .asp files. To see how these ASP pages will be displayed to the user, right-click the file Form.asp or List.asp, and then click **Preview in Browser**.

*filename*Form.asp

The Form page displays one record at a time in Form view. If you selected editing options when you ran the Data Form Wizard, Form view will provide command buttons that you can use to update, insert, delete, and filter records.

*filename*List.asp

The List page displays all of the records returned by the query in table format. It includes commands that enable a user to scroll forward and back through the list of records.

In List view, the Data Form Wizard will automatically add hyperlinks to each record. In Form view, the hyperlinks will enable users to jump to a single record.

*filename*Action.asp

In the data form, the Action page confirms a user action.

For example, if a user uses the Form page to delete a record, the Action page will be displayed with a confirmation message that indicates which record has been deleted.

You define a data source by creating a data source name (DSN), which is a logical name used by an ODBC database. When a Web application connects to a database through ODBC, it uses the DSN to access the data. The DSN identifies the database driver, network location, and other required data-access information.

You must define a data source name on your Web server and on your development computer. Visual InterDev uses the data source name on your development computer at design time to establish a connection to a database. Your Web application uses the data source name on your Web server when a user accesses a Web page.

To see a demonstration of how to define a data source, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

u **To create a data source name on your computer**

1. Open the Control Panel.
2. Start the ODBC utility.
3. Click the **System DSN** tab, and then click **Add**.

   A list of installed ODBC drivers will be displayed, such as Microsoft Access and Microsoft SQL Server. To install a new driver, you must use the Setup program for the driver you want to add.
   {ewc mvimg, mvimage,!tip.bmp}
4. In the **Create New Data Source** dialog box, select the driver you want to use for the database, and then click **Finish**.
5. In the **Setup** dialog box for the driver you have selected, add the appropriate information to define the data source.

Note Different drivers supply different dialog boxes for specifying information. For example, for a Microsoft SQL Server driver, you provide a server name, and then click the **Options** button to enter a database name.

Visual InterDev includes two design-time data access controls: a header and a footer. You use these controls as a pair to define the records you want to retrieve from a database.

The controls generate server-side script that runs when a user requests a page. The script queries the database, and returns a set of records to the user as HTML text.

In this section, you will learn how to use the data range controls to create a Web page for users to view the records of a database.

This section includes the following topics:

The last step in using the design-time ActiveX data range controls is to insert the **Data Range Footer** control.

To insert the Data Range Footer control

1. Just after the script you added to display the records from a recordset, right-click the .asp file, and then click **Insert ActiveX control**.
2. On the **Design-Time** tab, click **Data Range Footer Control**.
3. You can accept the default properties of the control.

Note To change the properties of a design-time control after inserting it, right-click the METADATA section of the .asp file, and then click **Edit Design Control**.

To use data range controls, you must first insert the **Data Range Header** control in an .asp file.

To insert the Data Range Header control

1. Open an .asp file.
2. Right-click the HTML section of the .asp file, and then click **Insert ActiveX Control**.
3. On the **Design Time** tab, select **Data Range Header** control, and then click OK.
4. On the **Control** tab, set **Data Connection** to the name of the project's data connection.
5. Set the command type for the query.
   a. If you select **Table**, **View**, or **Stored Procedure** as the command type, click the drop-down list box for the **Command Text** field, and then select the appropriate table, view, or stored procedure.
   b. If you select **SQL Command**, click **SQL Builder** to run the Query Designer.
   c. Use the Query Designer to define a query, and then close and save the query.

   For information about how to use parameter queries with the data range controls, go to the Microsoft Visual InterDev Samples Web site by clicking this icon.
   {ewc mvimg, mvimage,!intjump.bmp}
6. Set the range type to **Text**, **Form**, or **Table**.

   If you want the records that are retrieved by the data range controls to be displayed in an HTML table or form, insert the tag <TABLE> or <FORM> before the **Data Range Header** control. Select the appropriate range type in the **Data Range Header** control.

   When you insert the **Data Range Footer** control, it will use the range type to determine how to close the range. If the range type is Table, the **Data Range Footer** control will insert the </TABLE> tag. If the range type is Form, the control will insert the </FORM> tag. If the range type is Text, an end tag will not be inserted.
7. To add navigation buttons to the page, click the **Record Paging** option.
8. Click **Copy Fields**.

   This will copy to the Clipboard the ASP script that is used to display the fields. After you add the **Data Range Header** control, you can paste the script from the Clipboard to display the fields.
9. On the **Advanced** tab, select a cursor type.

   The **Data Range Header** control uses a cursor to retrieve records from a database. The type of cursor you select determines whether or not the control can page through records.

   The Keyset and Static cursor types are the only cursors that will allow record paging. The cursor types Forward Only and Dynamic will not support paging.

Note There are additional speed and capability differences between the cursor types. For information about these differences, see Article #Q149054 Choosing a rdoResultset CursorType from the Knowledge Base, which is available in the Library on this CD-ROM.

---

10. On the **Advanced** tab, select **Caching** or **No Caching** for the recordset.

    If you select **Caching**, the recordset will be stored in the **Session** object. When a user requests a new page of records, the data will be retrieved from the recordset on the Web server. This solution provides faster browsing, but requires significant overhead on the Web server and may decrease performance if you have many users.

    If you select **No Caching**, the current record location will be stored in the **Session** object. When a user requests a new page of records, the database will be queried again. A recordset will be created, the data retrieved, and the recordset will be closed. The browse time in this solution may be longer, however, it requires little overhead on the Web server and will allow more users to access your Web server.
11. Close the Data Range Header window.

    The **Data Range Header** control will generate the HTML tags and ASP script needed to retrieve the appropriate records. The control adds text surrounded by comments to the .asp file, as shown in the following example script:

```
<!--METADATA TYPE="DesignerControl" startspan
    ...
```

```
<!--METADATA TYPE="DesignerControl" endspan-->
```

Note You can change the properties of a design-time control after inserting it by right-clicking the METADATA section of the .asp file, and then clicking **Edit Design Control**.

After you have inserted the **Data Range Header** control, you can display fields from the recordset by adding script code to the .asp file.

The following example code displays the ClassID and Title fields from a **Data Range Header** control named DataRangeHdr1.

```
<TD> <%= DataRangeHdr1("ClassID") %></TD>
<TD> <%= DataRangeHdr1("Title") %></TD>
```

The HTML tags <TD> and </TD> cause each field to appear in a separate column of a table.

Note If you clicked **Copy Fields** on the **Control** tab of the **Properties** dialog box for the **Data Range Header** control, you can paste the code from the Clipboard into the .asp file.

If you did not click **Copy Fields**, you can type the code manually. You can also right-click the METADATA section of the **Data Range Header** control to redisplay the **Properties** dialog box, and then click **Copy Fields**.

To run a query, click the **Run** button on the **Query Designer** toolbar. The query will run, and the results will be returned in the Results pane of the Query Designer.

To check the syntax of an SQL statement without running a query, click the **Verify SQL Syntax** button on the **Query Designer** toolbar.

To run a stored procedure, right-click the name of the stored procedure in Data View, and then click **Run**. If the procedure requires arguments, the **Run Stored Procedure** dialog box will be displayed for you to enter values for each argument.

To see an illustration of the **Run Stored Procedure** dialog box, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

To complete the lab exercises in this chapter, you must have the required software. For detailed information about the labs and setup for the labs, see Labs in this course.

For background information on this lab, click each of these topics:

Objectives

Prerequisities

Lab Setup

To see a demonstration of the lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

To see a diagram of how the files you edit in this lab will fit into the State University Web site, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Estimated time to complete this lab: **60 minutes**

Note There are project and solution files associated with each lab. If you installed the labs during Setup, these files are in the folder <*Install Folder*>\Labs\Lab03 on your hard disk. If you did not install the labs during Setup, you can find the files in the \Labs\Lab03 folder of the *Mastering Web Site Development* CD-ROM.

Exercises

In the following exercises, you will practice working with the concepts and techniques covered in Chapter 3: Using Visual InterDev Data Tools.

Exercise 1: Setting Up the Database

In this exercise, you will set up the State University SQL Server database to work with the database labs.

Exercise 2: Adding a Data Connection

In this exercise, you will add a data connection to the State University database from your Web project.

Exercise 3: Running the Data Form Wizard

In this exercise, you will run the Data Form Wizard to create Web pages that display the Classes table in List view and Form view.

Exercise 4: Using the Data Range Controls

In this exercise, you will insert the **Data Range Header** and **Data Range Footer** controls to retrieve classes, majors, and descriptions.

After completing this lab, you will be able to:

- Add a data connection to a Web project.
- Use the Data Form Wizard to create Web pages that can retrieve and update data.
- Use the design-time controls **Data Range Header** and **Data Range Footer** to create data-driven HTML content.
- Use the Query Designer to build an SQL query, and insert the query into the **Data Range Header** control.

Before completing this lab you must be able to:

® Create Web pages, .asp files, and projects in Microsoft Visual InterDev.

® Write simple, server-side script, as described in Chapter 2: Developing a Web Project.

To complete this lab, you need the following:

® The Visual InterDev State University Web project.

The first step to creating data-driven Web pages is to add a data connection to your Web project. A data connection provides access to a data source from the project.

For information about adding a data connection, see Adding a Data Connection.

To see an illustration of how the Visual InterDev project should look after adding a data connection, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

u **Add a data connection**

1. In Visual InterDev, open the StateU project.
2. Right-click the project, and then click **Add Data Connection**.
3. On the **Machine Data Source** tab, select the StateU data source.
4. Type the name **StateU** for the data connection.
5. Set the run-time login to **sa**, with no password, and accept all other default values in the **Data Connection Properties** dialog box.

Note If you set up the database with a different login and a password, enter that login and password instead.

In this exercise, you will run the Data Form Wizard to create Web pages from which users can view and update the Classes table of the State University database.

For information about how to run the Data Form Wizard, see Using the Data Form Wizard.

To see an illustration of how the completed Web page will look, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Start the Data Form Wizard

1. In Visual InterDev, open the State University project.
2. To create a new file for the project, click **New** on the **File** menu.
3. In the **New File** dialog box, click the **File Wizards** tab.
4. Select **Data Form Wizard**, type **class** for the file name, and then click OK.

Complete the steps in the Data Form Wizard

1. Set the database connection to **StateU**.
2. Set the title at the top of the form to **Class List**.
3. Set the database object type to **Table**.
4. Select all fields from the Classes table in this order: **ClassID**, **Title**, **MajorID**, **Seats**, and **StartDate**.
5. Select the options **No, users can only browse information** and **Allow information to be filtered**.
6. In the **Viewing Options** dialog box, keep the default entries.
7. To specify how the Web pages should look, select the **Grid** theme.
8. To create the new Web pages, click **Finish**.

Test the results

1. Use the **Preview in Browser** option to open the classform.asp file created by the Data Form Wizard.
2. In Form view, click **List View** to see all of the classes in the database.

For an introduction to this chapter, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

Web sites are becoming more than just a means for publishing information. They have become client/server applications that use resources on both the client computer and the server computer.

In this chapter, you will learn how to plan and create a Web site by using a service-based application model. You will also learn about the members of a Web site development team, and the development tools available to them.

Objectives

By the end of this chapter, you will be able to:

® List and define the three types of services used for building a Web site.

® Describe four benefits of using a service-based application model.

® List the responsibilities of each member of a Web site development team.

® List and describe the products used by each Web site development team member.

For an introduction to this chapter, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

In this chapter, you will learn how to use Microsoft Visual InterDev to build and manage Web sites. You will also learn how to author static HTML pages by using Microsoft FrontPage. Static Web pages contain information that is not interactive.

Later in this chapter, you will learn how to create active content with Active Server Pages, and how to use Visual InterDev to create an .asp file.

Finally, you will learn how to use design-time controls to process server-side script.

The following illustration highlights the technologies you will learn about in this chapter.

{ewc MVIMG, MVIMAGE,!W02g460.bmp}

Objectives

After completing this chapter, you will be able to:

® Create a Web site with Microsoft Visual InterDev.

® Author static HTML pages with Microsoft FrontPage.

® Create HTML forms.

® Create Active Server Pages with Visual InterDev.

® Implement server-side script.

® Use design-time controls.

For an introduction to this chapter, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

In this chapter, you will learn how to use Microsoft Visual InterDev and FrontPage to create interactive Web pages by adding objects with ActiveX controls and Java applets. You will learn how Internet Explorer downloads and runs these objects to exchange data with a Web server.

You will also learn how to use controls that have been signed or marked by the author for security purposes.

The following illustration highlights the technologies you will learn about in this chapter.

{ewc MVIMG, MVIMAGE,!w04g110.bmp}

Objectives

After completing this chapter, you will be able to:

® Describe the differences between ActiveX controls and Java applets.

® Insert an ActiveX control and a Java applet on a Web page.

® Add a licensed ActiveX control to an HTML page.

® Create an HTML Layout that contains ActiveX controls, and add it to a Web page.

® Describe the implications of users setting the safety level in Microsoft Internet Explorer.

® Describe the purpose of downloading code with .cab and .inf files.

Objects such as ActiveX controls and Java applets play an important role in helping to make your Web content interactive. On an HTML page, both of these types of objects provide a user interface.

There are hundreds of controls and applets that you can use in your Web application. Once you identify the functionality you want, you can select the object that provides that functionality, and then simply drop the ActiveX control or Java applet onto your Web page.

In this section, you will learn about how Microsoft Internet Explorer downloads and runs ActiveX controls and Java applets. You will also learn about the differences between these two types of objects.

This section includes the following topics:

An ActiveX control is a binary file in an executable format. The <OBJECT> tag is a reference to the binary file. For example, if you insert a **Calendar** control with the <OBJECT> tag, the control must be installed on a user's computer before it can be viewed.

In this section, you will learn how to use ActiveX controls on a Web page by using the <OBJECT> tag. You will also learn about the <PARAM> tag, which you can use to set initial object properties for a control.

This section includes the following topics:

You can add and edit an ActiveX control either with the Visual InterDev Source Editor or the FrontPage Editor. In this topic, you will learn how to add and edit an ActiveX control with Visual InterDev.

To add an ActiveX control with the Visual InterDev Source Editor

1. In the Source Editor window for a Web page, right-click the location where you want to insert the control, and then click **Insert ActiveX Control**.

   You must click between the beginning and ending <BODY> tags for the page.
2. In the **Insert ActiveX Control** dialog box, on the **Controls** tab or **Design-time** tab, select the control you want to add.

   Visual InterDev starts the Object Editor in the HTML Source Editor window, and then opens the **Properties** dialog box for the control.
3. In the **Properties** dialog box, set initial values for the object properties.

   As you change the values of properties, the Object Editor updates and displays the changes.
4. Type a location for the **CodeBase** property.

   The **CodeBase** property specifies an Internet location for the source files.
5. Close the Object Editor.

   When you close the Object Editor, it inserts an <OBJECT> tag for the control you want to add to the Web page.

To edit an object with the Visual InterDev Source Editor

1. In the Source Editor window for the HTML page or .asp page, right-click anywhere between the <OBJECT> tags that contains the object you want to edit, and then click **Edit ActiveX Control**.

   Visual InterDev starts the Object Editor with the selected ActiveX control.
2. With the Object Editor, edit the properties of the ActiveX control.
3. Close the Object Editor.

   When you close the Object Editor, it updates the <OBJECT> tag for the control to reflect your edits.

To see a demonstration of how to use Visual InterDev to add and edit an ActiveX control on an HTML page, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

For an introduction to this chapter, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

In this chapter, you will learn how to access a database from a Web page by working with the Data Form Wizard and the Visual InterDev data range controls. You will also learn how to use the Query Designer to create SQL queries for a database.

The advantage of using the wizard and the design-time data range controls is that you do not need to know how to write server-side script to perform database functions. The wizard and controls generate the script for you.

To create more advanced Web pages that access and query a database, you can write script in a Web page without using the wizard. In Chapter 7: Creating Database-Aware Web Pages, you will learn how to access a database by writing server-side script that calls ActiveX Data Objects (ADO).

The following illustration highlights the technologies you will learn about in this chapter.

{ewc MVIMG, MVIMAGE,!w03G090.bmp}

Objectives

By the end of this chapter, you will be able to:

® Add a data connection to a Web project.

® Use the Query Designer to create SQL queries.

® Use the Data Form Wizard to generate Web pages that enable a user to view, add, update, and delete data from a database.

® Use the design-time data range controls to create Web pages that display records from a database.

For an introduction to this chapter, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

In this chapter, you will learn how to add client-side script to a Web page. With client-side script, you can create active Web pages that respond to user input, validate data in a form, and manipulate objects, such as ActiveX controls and Java applets.

In the previous chapter, you learned how to add objects to a Web page. In this chapter, you will learn how to create event procedures for objects, and access properties and invoke methods of objects at run time.

The following illustration highlights the technologies you will learn about in this chapter.

{ewc MVIMG, MVIMAGE,!w05g310.bmp}

Objectives

After completing this chapter, you will be able to:

® Describe the difference between client-side script and server-side script.

® Describe the difference between Microsoft Visual Basic Scripting Edition (VBScript) and JavaScript.

® Use VBScript to create event procedures for standard HTML controls and ActiveX controls.

® Use the HTML object model in VBScript.

® Add page-initialization script to a Web page.

® Add VBScript to validate data in a form.

® Handle run-time errors in VBScript.

® Use the Visual InterDev Script Wizard to create event procedures for ActiveX controls and standard HTML controls.

® Use the Microsoft Script Debugger to debug script errors.

For an introduction to this chapter, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

In this chapter, you will learn how to use Active Server Pages in your Web application. In the previous chapter, you learned how to add client-side script to a Web page. This chapter describes how to add server-side script to manipulate objects on a Web server and create a Web application.

Active Server Pages provides a server-side scripting environment that enables you to perform such tasks as reading information from an HTTP request, customizing the HTTP response, storing information about a user, and extracting the capabilities of the user's browser.

This illustration highlights the technologies you will learn about in this chapter.

{ewc MVIMG, MVIMAGE,!w06g030.bmp}

Objectives

After completing this chapter, you will be able to:

® Describe the HTTP protocol.

® List and describe the Active Server Pages objects.

® Create and use cookies.

® Use the **Request** and **Response** objects to dynamically change the HTTP response.

® Process form data with server-side scripting.

® Save session-specific information by using the **Session** object.

® Save application-specific information by using the **Application** object.

® Use an ActiveX server component in your Web application.

For an introduction to this chapter, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

In this chapter, you will learn how to create Web pages that retrieve and update information in a database by using the ActiveX Data Objects (ADO) and the Advanced Data Connector (ADC).

In Chapter 3: Using Visual InterDev Data Tools, you learned how to use the Data Form Wizard to create Web pages containing ADO code. In this chapter, you will learn how to write ADO code. Writing ADO code provides more flexibility than using the Data Form Wizard to generate the code. You can use ADO with various host applications, including Active Server Pages, Visual Basic, and Visual C++.

By using the Advanced Data Connector, you can create a Web page that retrieves a set of records from a database and caches the data locally on a user's computer. You set attributes of the ADC control to display the data to the user from the cached recordset in data-bound ActiveX controls on the Web page.

Because the data is cached locally, a user can view, edit, and delete records without accessing the server again. A user can modify several records and then submit the entire recordset to the server for processing.

The following illustration highlights the technologies you will learn about in this chapter.

{ewc MVIMG, MVIMAGE,!w07g070.bmp}

Objectives

By the end of this chapter, you will be able to:

® Describe the differences between ActiveX Data Objects (ADO) and the Advanced Data Connector (ADC).

® Use ActiveX Data Objects to retrieve and update data and to handle errors returned from a database provider.

® Bind the data from the **AdvancedDataControl** object to data-bound controls on a Web page.

® Use the **AdvancedDataSpace** and **AdvancedDataFactory** controls to create a recordset object on the client computer.

For an introduction to this chapter, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

In Chapter 8: Creating ActiveX Server Components, you learned how to build ActiveX server components that implement business rules for business processes.

In this chapter, you will learn about Microsoft Transaction Server (MTS), an application that provides transaction and resource management for ActiveX server components. You will also learn how to create Microsoft Transaction Server components, which are ActiveX server components that work within the MTS architecture.

The following illustration highlights the technologies you will learn about in this chapter.

{ewc MVIMG, MVIMAGE,!w09g080.bmp}

Objectives

After completing this chapter, you will be able to:

® Explain what a transaction is and why it must conform to the transaction ACID test.

® Explain how the two-phase commit process works.

® Identify the components of the Microsoft Transaction Server architecture.

® Describe the role of the **Context** object.

For an introduction to this chapter, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

In this chapter, you will learn how to use Microsoft Internet Information Server (IIS) and Microsoft Windows NT to control access to your Web site.

Microsoft Internet Explorer and IIS provide features that you can use to limit which users can access your Web site. You can specify which files a user can access and what operations the user can perform on your Web server. You will also learn how to use digital certificates to identity users. Finally, you will learn how to use the Secure Sockets Layer (SSL) protocol to encrypt data during transmission.

The following illustration highlights the technologies you will learn about in this chapter.

{ewc MVIMG, MVIMAGE,!w10g050.bmp}

Objectives

After completing this chapter, you will be able to:

® List the Microsoft Windows NT and SQL Server logon process that occurs when a user requests a Web page on a Microsoft Internet Information Server.

® Use the Windows NT File System (NTFS) to assign permissions to folders and files.

® Configure a Web server to allow or prevent anonymous access.

® Set permissions for the Anonymous user account.

® List the logon process that occurs when an Active Server Page contains script that accesses a Microsoft SQL Server database.

® Describe the purpose of digital certificates.

The labs for this course require the State University database, which you will set up in this exercise.

The database is provided in both Microsoft Access and Microsoft SQL Server formats. You can use either database format to complete Labs 3 through 8. However, if you are working on a computer that doesn't have the disk space or power to run SQL Server 6.5, you will need to use the Microsoft Access database.

To complete Lab 9, you must use the Microsoft SQL Server database. Lab 9 uses Microsoft Transaction Server, which works only with databases that support the Microsoft Distributed Transaction Coordinator. If you cannot use Microsoft SQL Server, you will not be able to complete Lab 9.

Install the Microsoft SQL Server State University database

If you plan to use the Microsoft SQL Server format of the database, complete the steps that follow the demonstration.

To see a demonstration of how to install the database, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

1. In an SQL Server 6.5 database, run the SQL Enterprise Manager from the Microsoft SQL Server 6.5 Programs folder. This folder can be accessed from the Windows NT Start menu.
2. To open the SQL Server within the SQL Enterprise Manager, click the plus sign (**+**) next to the server name.

   If your server does not appear, you will need to register your server in the SQL Enterprise Manager. To do this, click **Register Server** on the **Server** menu, select the server, and then type a valid login and password.

   If your server does not start, right-click the traffic signal and then click **Start**.
3. To create a database device for the StateU Web project, right-click the Database Devices folder, and then click **New Device**.
4. To name the new device, type **StateU** in the **Name** field.
5. In the **Size** field, type **5 MB**.
6. To create the StateU database device, click **Create Now**.
7. To create the database, right-click the Databases folder, and then click **New Database**.
8. To name the new database, type **StateU** in the **Name** field.
9. In the **Data Device** field, select the StateU device.
10. In the **Size** field, type **5 MB**.
11. To create the StateU database, click **Create Now**.
12. In the SQL Enterprise Manager, click **SQL Query Tool** on the **Tools** menu.
13. Click the **Load SQL Script** button on the toolbar.
14. Select the StateU.sql file from the folder C:\Mwd\Labs\Lab03, and then click **Open**.
15. To run the SQL script, click **Execute** on the **Query** menu.

    This script creates tables, and adds data to the tables. The script is complete when the **Execute Query** button changes from a black arrow to a green arrow.
16. Close the Query Tool window, and then click **Yes** when prompted if the window should be closed.

    To see an illustration of how the SQL Enterprise Manager should look after creating the State University database, click this icon.
    {ewc mvimg, mvimage,!illust.bmp}

Install the Microsoft Access State University database

If you plan to use the Microsoft Access format of the database, complete the following steps:

1. Create a folder on your Web server named C:\Database.
2. Use Windows Networking to share this folder with other computers on the network.
   a. To share the folder, right-click the Database folder in Windows Explorer, and then click **Sharing**.
   b. Name the share **Database** so it is available to Visual InterDev development workstations as *Web server name*\database.

3. Copy the file C:\Mwd\Labs\Lab03\StateU.mdb from your development workstation to the Database folder on your Web server.

Note Verify that the StateU.mdb file is not read-only. If it is read-only, you must change it to read-write before you can update the data.

---

u Set up the ODBC system DSN

You must create a data source name on your Web server and on your Visual InterDev development computer. For information about creating a data source name, see Creating an ODBC Data Source Name.

On your development computer and on your Web server, complete the following steps:

1. Open the Windows Control Panel.
2. Open the ODBC Data Source Administrator.
3. On the **System DSN** tab, click **Add**.
4. If you are using Microsoft SQL Server:
   a. Double-click **SQL Server**.
   b. For the data source name, enter **StateU**.
   c. In the **Server** field, enter your SQL Server name. If your SQL Server is on the same computer as your Web server, select **(local)** for the server.
   d. Click the **Options** button.
   e. Type **StateU** for the database name, and then click OK.
5. If you are using Microsoft Access:
   a. Double-click **Microsoft Access Driver (.mdb)**.
   b. Type **StateU** for the data source name.
   c. Click **Select**.
   d. In the **Database Name** field, type *Web server name*\database\stateu.mdb, and then click OK.

      Make sure that the **Directories** list in the **Select Database** dialog box shows the folder *Web server name*\database, and not a network drive letter, such as E:\.

      If you have a network drive letter mapped to the share *Web server name*\database, and you do not type the complete server and share name when selecting the database, the ODBC **Select** dialog box will default to the mapped network drive letter and cause problems in later labs.
6. To close the ODBC Data Source Administrator utility, click the OK button again.

   To see an illustration of how the ODBC Data Source Administrator should look after adding the StateU data source, click this icon.
   {ewc mvimg, mvimage,!illust.bmp}

State University has created a fun Java applet that animates their mascot with flips. In this exercise, you will add this applet to the mascot.htm page in the State University Web site.

In later labs, you will implement the user interface to start and stop the mascot, and to change the tumbling speed.

To see an illustration of how your Web page should look, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

u Add the mascot Java applet with the FrontPage Editor

1. In Visual InterDev, add the Java class file \MWD\Labs\Lab04\mascot.class to the controls folder of the StateU project.
2. Using the FrontPage Editor, open the file mascot.htm.
3. To insert a Java applet at the location of the placeholder text, click **Other Component** on the **Insert** menu, and then click **Java Applet**.

   The **Java Applet Properties** dialog box displays.

   a. Set **Applet Source** to mascot.class.

   b. Set **Applet Base URL** to controls.

   c. Set **Width** to 280 and **Height** to 180.

   d. Add the following parameters with the specified values.

Name	Value	Purpose
speed	50	Initial speed of flips
direction	1	Flip right to left

   e. To close the **Java Applet Properties** dialog box, click OK.
4. To see the <APPLET> tag added to mascot.htm, view the HTML source code of the page, as shown in the following example code:

```
<applet
    code="mascot.class"
    codebase="controls"
    align="baseline"
    width="280"
    height="180" >
    <param name="direction" value="1">
    <param name="speed" value="50">
</applet>
```

5. Save your changes, and then close the FrontPage Editor.

u Test the Web page

® In the Visual InterDev InfoViewer, preview the mascot Web page.

Some of the ActiveX controls you add to a Web page may require a run-time license. In this exercise, you will add a license file for the **Spin Button** control to the StateU Web project, and then add tags to the mascot Web page for the Licensed Control Manager.

u View mascot.htm without a license

There is an unsigned **Spin Button** control on the page mascot.htm that requires a run-time license.

1. In Microsoft Internet Explorer, set the Safety Level to Medium or None so you can view the unsigned control.
   a. In Microsoft Internet Explorer, click the **View** menu, and then click **Options**.
   b. On the **Security** tab, click **Safety Level**.
   c. In the **Safety Level** dialog box, set the Safety Level to Medium or None.
2. In Microsoft Internet Explorer, go to the page mascot.htm in the StateU Web site.

   If you do not have a licensed copy of the **Spin Button** control installed on your computer, Microsoft Internet Explorer will display the control, but it will be inactive.

   To see an illustration of how the mascot.htm Web page should look without a license for the **Spin Button** control, click this icon.
   {ewc mvimg, mvimage,!illust.bmp}

u Add the License Manager to the mascot page

1. In Visual InterDev, add the file mascot.lpk from the folder \Labs\Lab04 to the StateU Web project in the same folder as mascot.htm.
2. Open the file mascot.htm with the Visual InterDev Source Editor.
3. After the <BODY> tag, add an <OBJECT> tag for the License Manger, and add a <PARAM> tag to set the **LPKPath** property to the file mascot.lpk, as shown in the following example code:

```
<OBJECT CLASSID="clsid:5220CB21-C88D-11CF-B347-00AA00A28331">
    <PARAM NAME="LPKPath" VALUE="mascot.lpk">
</OBJECT>
```

   For information about the License Manager, see Using Licensed Controls.
4. Save your changes, and then preview the file mascot.htm in the Visual InterDev InfoViewer.

   If the **Spin Button** control is still inactive, you will need to reload the control by restarting all running Web browsers and Visual InterDev.

To complete the lab exercises in this chapter, you must have the required software. For detailed information about the labs and setup for the labs, see Labs in this course.

For background information on this lab, click each of these topics:

Objectives

Prerequisites

Lab Setup

To see a demonstration of the lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

To see a diagram of how the files you edit in this lab will fit into the State University Web site, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Estimated time to complete this lab: **60 minutes**

Note There are project and solution files associated with this lab. If you installed the labs during Setup, these files are in the folder <*Install Folder*>\Labs\Lab05 on your hard disk. If you did not install the labs during Setup, you can find them in the \Labs\Lab05 folder of the *Mastering Web Site Development* CD-ROM.

Exercises

The following exercises provide practice working with the concepts and techniques covered in Chapter 5: Adding Client-Side Script.

Exercise 1: Creating Event Procedures

In this exercise, you will write script to automate the State University mascot Java applet.

Exercise 2: Editing Hyperlinks

In this exercise, you will change two frames by adding code to a hyperlink in the links.htm Web page.

Exercise 3: Validating a Form

In this exercise, you will add code to a form to validate user-entered data on the form.

Exercise 4 (Optional): Combining Client and Server Script

In this exercise, you will add code to read information from the StateU database, and add items to the pop-up menus on the page home.htm.

After completing this lab, you will be able to:

® Add client-side script to a Web page that interacts with ActiveX controls and Java applets.

® Validate controls on a form before submitting the form to the Web server.

® Write script for a hyperlink to change the source files for two frames.

® Combine client-side and server-side script on a Web page.

Before completing this lab you must be able to:

- Add files to a Visual InterDev project.
- Edit files with the Visual InterDev Source Editor.
- Create a form and set the ACTION attribute.
- Add the **Data Range Header** and **Data Range Footer** design-time controls to read information from a database, as described in Chapter 3: Using Visual InterDev Data Tools.

To complete this lab, you need the following:

- The StateU Visual InterDev Web project.
- These files in the project: mascot.htm, mascot.lpk, links.htm, default.htm, and home.htm.
- The ActiveX controls **Image**, **Spin**, and **Pop-up Menu**.
- The mascot Java applet.
- These image (.gif) files in the images folder of the StateU Web project: bullet, history, math, music, sulogo, sutitle, mascot, mascotlogo, and img0001.gif–img0018.gif.

To complete the lab exercises in this chapter, you must have the required software. For detailed information about the labs and setup for the labs, see Labs in this course.

For background information on this lab, click each of these topics:

Objectives

Prerequisites

Lab Setup

To see a demonstration of the lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

To see a diagram of how the files you edit in this lab will fit into the State University Web site, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Estimated time to complete this lab: **60 minutes**

Note There are project and solution files associated with this lab. If you installed the labs during Setup, these files are in the folder *<Install Folder>*\Labs\Lab06 on your hard disk. If you did not install the labs during Setup, you can find them in the \Labs\Lab06 folder of the *Mastering Web Site Development* CD-ROM.

Exercises

The following exercises provide practice working with the concepts and techniques covered in Chapter 6: Using Active Server Pages.

Exercise 1: Reading Form Data

In this exercise, you will create an Active Server Page that reads data supplied by an HTML form and saves it in the **Session** object.

Exercise 2: Starting a Session

In this exercise, you will write an event procedure for the **Session** object that routes all users to the Profile page before they can access the State University Web site.

Exercise 3 (Optional): Implementing a Hit Counter

In this exercise, you will add code to the global.asa file that saves the number of users who have accessed the State University Web site, and writes the number into a text file when the Web server shuts down.

After completing this lab, you will be able to:

- Read form data.
- Save information in the **Session** object.
- Add code to the Session_OnStart event procedure that routes all users to the Profile page when starting a session.
- Write application-level information into a text file.

Before completing this lab you must be able to:

- Add files to a Visual InterDev project.
- Edit files with the Visual InterDev Source Editor.
- Create an Active Server Page and add server-side script, as described in Chapter 2: Developing a Web Project.

To complete this lab, you need the following:

® The Visual InterDev StateU Web project.

® The file profile.htm.

Visual InterDev provides the Query Designer, which enables you to build and run SQL queries on a database as you develop your Web application.

You can create a query by selecting tables and fields in the diagram pane, or you can type an SQL statement directly into the SQL pane of the Query Designer.

The **Query Designer** toolbar is displayed when you open a table in Data View. With the **Query Designer** toolbar, you can display the Diagram, Grid, SQL, and Results panes to view and modify a query.
{ewc mvimg, mvimage,!tip.bmp}

To see an illustration of the Query Designer with each of its panes, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

The Query Designer panes all relate to one another. When you move the focus from one pane to another, the Query Designer updates all of the panes to reflect your query.

In this section, you will learn how to create and run an SQL query.

This section includes the following topics:

® Selecting Tables and Columns

® Setting Criteria and Sort Order

® Running a Query

With the Query Designer in Visual InterDev, you can specify which tables to include in a query and which columns to display.

To see a demonstration of how to create and run a query, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Adding Tables

To add a table to a query, drag the table from Data View to the Diagram pane of the Query Designer.

If the tables are related, the Query Designer will automatically display a join line to indicate the relationship. To modify the type of relationship, right-click the join line, and then click **Properties**.

To see an illustration of the **Properties** dialog box for a join line, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Displaying Columns

To specify which columns you want to include in your query, select the columns in the Diagram pane of the Query Designer.

If you want to include a column in the query, but not display the column in the query results, clear the **Output** check box in the Grid pane, and then enter the criteria in the appropriate column.
{ewc mvimg, mvimage,!tip.bmp}

You can set criteria and specify a sort order for a query in the Grid or SQL panes of the Query Designer.

Setting Criteria

To set criteria for a query, type the expression in the **Criteria** column for the appropriate field in the Grid pane.

The following table provides some example criteria expressions and the resulting records.

Criteria	Records found
>'M'	Records in which the criteria field is greater than M.
Like 'M%'	Records in which the criteria field begins with an M.
>=5 and <10	Records in which the criteria field is greater than or equal to 5 and less than 10.

{ewc mvimg, mvimage,!tip.bmp}

Setting Sort Order

To sort the records of a query, click **Ascending** or **Descending** in the **Sort Type** column. If you sort by multiple columns, you can set the **Sort Order** column to specify the order in which the sort is performed.

For example, to sort by last name and then by first name, set the sort order for last name to 1 and set the sort order for first name to 2.

Students at State University need a list of all offered classes and the classes available for each major. Because this information is constantly changing, you will create a Web page with a table that retrieves data from the database by using the **Data Range Header** and **Data Range Footer** controls.

To see an illustration of how the completed Web page should look, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

In this exercise, you will create a table header that contains column titles for each column that is returned from the database. You will then insert the **Data Range Header** control and build a query that retrieves the classes, majors, and descriptions of majors. Finally, you will create the HTML tags that display records from the database, and then add the **Data Range Footer** control.

For information about how to use data range controls, see Using the Data Range Controls.

Create a table header

1. Add the file class_descriptions.asp from the folder \MWD\Labs\Lab03 to the root folder of the Web site.
2. In the class_descriptions.asp file, just above the </BODY> tag, create a table with one row.

   This will be the header row for all returned values. The header row will contain headers for four columns: Class ID, Title, Major ID, and Description, as shown in the following example code:

```
<TABLE border=0 cellpadding=3>
<TR bgcolor=silver><B>
   <TD>Class ID</TD>
   <TD>Title</TD>
   <TD>Major ID</TD>
   <TD>Description</TD></B>
</TR>
```

Note In the next procedures, additional rows will be added to the table. Do not close the table with the ending </TABLE> tag. The **Data Range Footer** control will add this tag automatically.

Insert the Data Range Header control

1. Just after the </TR> tag in the script, insert the **Data Range Header** control.
2. Set the ID of the **Data Range Header** control to ClassMajors.
3. Set the data connection to StateU.
4. Set the range type to Table.

Note Selecting **Table** as the range type will cause the **Data Range Footer** control to insert the </TABLE> tag and display navigation buttons after the table.

---

5. Select the **Advanced** tab, and set the cursor type to Keyset.
6. On the **Control** tab, select **Record Paging** for a page size of 10.
7. To display the Query Builder, click **SQL Builder**.
8. Drag the Classes and Majors tables from the Data View pane of the project workspace to the Design pane of the SQL Builder.

   Query Builder will recognize the relationship between the tables automatically, and will add a join line between them.
9. In the Classes table, select the ClassID and Title fields.
10. In the Majors table, select the MajorID and Description fields.
11. Sort the query on the Title field in ascending order.

    To see an illustration of how the completed Query Builder should look, click this icon.
    {ewc mvimg, mvimage,!illust.bmp}
12. Close the Query Builder, save the results, and then close the Object Editor.

The script for the **Data Range Header** control will be inserted in your Web page.

u **Add HTML tags to display records**

1. Just after the **Data Range Header** script, add HTML code that creates a new table row with four columns.

   Fill in the four columns with the four data fields from the **ClassMajors** control: ClassID, Title, MajorID, and Description, as shown in the following code:

```
<TR bgcolor=white>
   <TD><%=ClassMajors("ClassID")%></TD>
   <TD><%=ClassMajors("Title")%></TD>
   <TD><%=ClassMajors("MajorID")%></TD>
   <TD><%=ClassMajors("Description")%></TD>
</TR>
```

u **Insert the Data Range Footer control**

1. Just before the </BODY> tag, insert the **Data Range Footer** control.

   You do not need to change any properties for this control.
2. Close the Property Sheet and the Control Design window.

u **Test the Results**

® Preview the Web page in the browser, and verify that it correctly returns the classes and the descriptions of the majors.

If the **Query** toolbar is hidden, you can display it by right-clicking the toolbar in Visual InterDev, and then clicking **Query**.

To create an SQL query that includes aggregate functions such as **Count** or **Max**, right-click the Diagram pane.

To add a **Group By** column to the Grid pane, click **Group By**. You can then select a function from the drop-down list box in the **Group By** column for the appropriate field.

When you add a data connection to your Web project, you can choose either a system DSN or a file DSN.

If you select a system DSN, the DSN name is referenced in the script generated by Visual InterDev in the Global.asa file. Additional information is retrieved from the registry at run time, based on the DSN name. You must create a DSN registry entry on the Web server and on your development computer.

If you select a file DSN, all of the information needed to make a data connection is copied from the .dsn file into the script that Visual InterDev generates in the Global.asa file. You no longer need the original .dsn file to connect to the database server, and the connection information travels with the Web application. Using this approach, it's easy to move your Web application from one computer to another because you don't have to recreate the DSN on each computer.

For more information about using data connections without a DSN, search for "DSN" in the Visual InterDev InfoViewer, and then double-click "Adding a Data Connection to a Web Project."

The data range controls provide more flexibility than the Data Form Wizard. You can use the data range controls to add data retrieval capabilities to an existing Web page, and to display the records in any format.

Data range controls do not include the edit and update capabilities provided by the Data Form Wizard. However, if you are familiar with how to use ActiveX Data Objects (ADO), you can modify the script that is generated by the data range controls, and add edit and update features.

The design-time ActiveX controls **Data Range Header** and **Data Range Footer** work as a pair to indicate the beginning and end of a code section that is repeated for each record of a recordset.

To see a demonstration of how to use data range controls, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

The **Data Range Header** control creates a recordset and starts a loop statement. The **Data Range Footer** control completes the loop statement and adds navigation buttons to the .asp file. Any HTML tags or ASP script between the data range header and footer will be repeated for each record in the recordset.

To see an illustration of a Web page that has been created with data range controls, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

For information about how to use ADO, see Chapter 7: Creating Database-Aware Web Pages.

u **To use data range controls on an Active Server Page**

1. Insert the **Data Range Header** control on an Active Server Page, and then define the SQL query used to retrieve records.
2. Add script to the Active Server Page to display fields from the recordset.
3. Insert the **Data Range Footer** control to specify the end of the repeated code section on the Active Server Page.

When you define the SQL query for the **Data Range Header** control, you can use parameters in the query. To use parameters, create an HTML form that retrieves a value from the user and passes it to the Active Server Page. This value can then be used by the design-time data range controls.

To see a demonstration of how to use a parameter in an SQL statement used by data range controls, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

In this section, you will learn how to use ADO objects to create a connection from a Web page to a data source, retrieve and modify data, run stored procedures, and handle errors.

This section includes the following topics:

Data access models represent the structure and data of a database as a set of objects. These models provide a standard interface for communicating with multiple types of databases.

Microsoft has introduced three data access models: Data Access Objects (DAO), Remote Data Objects (RDO), and ActiveX Data Objects (ADO). Eventually, ADO will include all of the features of DAO and RDO and will replace the previous two models as the standard for accessing data.

For a discussion of data access models, click this icon.
{ewc mvimg, mvimage,!exppov.bmp}

The following table provides an overview of the three data access models.

Data access model	Introduced with	Features
Data Access Objects (DAO)	Microsoft Access 1.0 and Visual Basic 3.0	Uses the Microsoft Jet database engine and ODBC to access remote databases.
Remote Data Objects (RDO)	Visual Basic 4.0	Object interface to ODBC API. Does not use the Microsoft Jet database engine.
ActiveX Data Objects (ADO)	Microsoft Internet Information Server (IIS) 3.0 and Active Server Pages	Has a simpler object hierarchy, can be used with any automation controller, and works with databases that provide either an ODBC driver or an OLE/DB provider.

In this section, you will learn how to use the Advanced Data Connector (ADC) to create Web pages that retrieve and update information in a database.

To install the Advanced Data Connector from the CD-ROM, see Advanced Data Connector in the Resources section of the Library. To install the ADC from a Web site, go to the Microsoft ADC Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

The Advanced Data Connector is a set of components that you can use to build Web applications for accessing ODBC-compliant databases. ADC provides a data-binding feature that binds the data from a recordset to data-aware ActiveX controls on a Web page.

To see an illustration of a Web page that uses an ActiveX control to display data in a grid on a Web page, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

To use ADC, you work with ActiveX controls, HTML, and client-side script. You add the **AdvancedDataControl** object to a Web page, and set attributes to indicate the data to retrieve and the controls in which to display the data.

You can also write client-side script to request a new set of records or submit changes to the database at run time.

This section includes the following topics:

1. When you add a data connection to a Visual InterDev project, how will your Web project be modified?

{ewc mvimg, mvimage,!answer.bmp}
A. Script will be added to the Global.asa file.

{ewc mvimg, mvimage,!answer.bmp}
B. The **Data Command** control will be added to all of your existing ASP pages.

{ewc mvimg, mvimage,!answer.bmp}
C. Script will be added to all existing ASP pages.

{ewc mvimg, mvimage,!answer.bmp}
D. Script will be added to the default ASP page.

2. What output is generated by the Data Form Wizard?

{ewc mvimg, mvimage,!answer.bmp}
A. A single .asp file that displays records in different views.

{ewc mvimg.mvimage,!answer.bmp}

B. As many as three .asp files, depending on your option selections.

{ewc mvimg.mvimage,!answer.bmp}

C. A single .htm file that displays records in different views.

{ewc mvimg.mvimage,!answer.bmp}

D. As many as three.htm files, depending on your option selections.

3. You have inserted the Data Range Header control, you have set the Paging option to enable the user to page through. Which type of cursor supports the ability to page through records?

{ewc mvimg.mvimage,!answer.bmp}

A. Only the Keyset cursor.

{ewc mvimg.mvimage,!answer.bmp}

B. Only the Dynamic cursor.

{ew

C. Either the Keyset or the Static cursor.

{ewc mvimg, mvimage,!answer.bmp}

D. Either the Dynamic or the Forward Only cursor.

{ewc mvimg, mvimage,!answer.bmp}

Database systems generally provide a driver that conforms to a standard interface. This enables you to access the database by using any technology that also supports the interface.

To gain access to a database from a Web page, you can use a variety of technologies, such as the Internet Database Connector, Advanced Data Connector, and ActiveX Data Objects.

Types of Database Interfaces

The following standard database interfaces are defined:

® Open Database Connectivity (ODBC)

ODBC is the standard interface to relational database systems. Most database systems provide an ODBC driver, which enables you to access a database from any tool that works with ODBC-compliant databases. This interface is used specifically for SQL databases. For more information about ODBC, go to the Microsoft ODBC Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

® OLE DB

OLE DB is a set of interfaces for universal data integration, regardless of the data type. For example, OLE DB can be used to access both SQL databases as well as any other type of database that provides an OLE DB driver.

For more information about OLE DB, go to the Microsoft OLE DB Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

Technologies to Access a Database

There are several technologies that you can use to access a database from a Web page.

® Internet Database Connector

With the Internet Database Connector (IDC), you can submit a query from a Web form to retrieve records from an ODBC-compliant database.

The IDC is a built-in feature of Microsoft Internet Information Server (IIS). You can use it to submit any SQL statement. IDC does not work with OLE DB databases.

® Advanced Data Connector

The Advanced Data Connector (ADC) is a set of ActiveX controls that you can place on a Web page to enable client-side access to a database. When ADC controls are downloaded with a Web page, they can retrieve, update, and create records on either an OLE DB database or an ODBC-compliant database.

ADC controls use the HTTP protocol to communicate with a Web server and retrieve and update database information.

The ADC keeps a recordset cache on the client system, which reduces the round trips to the server as the user scrolls through the recordset. This control also works with data-bound controls to easily display information from a recordset.

For more information about ADC controls, see Chapter 7: Creating Database-Aware Web Pages.

® ActiveX Data Objects (ADO)

ADO is the next generation technology of data access models. ADO is a collection of Automation objects that can retrieve, update, and create records in any OLE DB or ODBC database. You can use ADO from any Automation controller, such as .asp files and ActiveX server components.

Because ADO uses Automation, you can use ADO in any application or program that is an Automation controller, such as Active Server Pages, Microsoft Internet Explorer, Visual Basic, and Visual C++.

ADO provides the most flexibility of the database access technologies. Because ADO runs on the Web server and returns straight HTML text, it can work with any browser.

For more information about ADO, see Chapter 7: Creating Database-Aware Web Pages.

```
<OBJECT ID="popClasses"
    CLASSID="CLSID:7823A620-9DD9-11CF-A662-00AA00C066D2"
    CODEBASE="controls/iemenu.ocx" >
</OBJECT>
```

```
Sub imgMath_Click()
    popClasses.PopUp
End Sub
```

An HTML Layout is like a drawing board on which you can add multiple ActiveX controls. With an HTML Layout, you can drag and drop a control to a specific location, size a control, and group and align controls. You can also place one control on top of another.

Each HTML Layout is saved as an ASCII text file with an .alx extension. The .alx file is independent of the HTML code.

To see an illustration of the relationship between the source HTML file and the .alx file, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

The .alx file contains the <OBJECT> tags for ActiveX controls. To display the HTML Layout on an HTML page, add a reference to the **HTML Layout** control, and specify the .alx file in a <PARAM> tag. This enables you to use the same HTML Layout for multiple HTML pages.

To see a demonstration of how to create and use an HTML Layout, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Creating HTML Layouts

There are two steps to creating an HTML Layout and incorporating it into HTML code. First, you create the HTML Layout and save it to a file. Then, you then add a reference to the HTML Layout in the .htm or .asp file so the layout will display in an HTML page at run time.

To create an HTML Layout

1. On the **File** menu, click **New**.
2. On the **Files** tab, click **HTML Layout**.
3. To add the HTML Layout to the current project, select the **Add to Project** check box.
4. In the **File Name** box, type a name for the layout.
5. In the **Location** box, verify that the path is correct, and then click OK.

   This starts the HTML Layout Editor for the new layout.
6. To specify the default properties for the HTML Layout (such as height, width, and background color), right-click anywhere in the HTML Layout Editor window, and then click **Properties**.
7. In the Toolbox, click each control you want to use, and then draw it on the HTML Layout.
8. To add additional ActiveX controls to the HTML Layout, click **ActiveX Control** on the **Insert** menu. Select a control, and then click OK.

   {ewc mvimg, mvimage,!tip.bmp}
9. To set properties to the controls you have placed on the layout, right-click each control, and then click **Properties**.
10. To save the HTML Layout, click **Save** on the **File** menu.

To insert an HTML Layout in a Web page

1. In Visual InterDev, create or open the Web page in which you want to insert the HTML Layout.
2. Right-click the Web page, and then click **Insert HTML Layout**.

   This displays the **URL Picker** dialog box.
3. In the **URL Picker** dialog box, select the HTML Layout that you want to insert, and then click OK.

   The Visual InterDev Source Editor inserts an <OBJECT> tag in the Web page to reference the .alx file.
   {ewc mvimg, mvimage,!tip.bmp}

Editing HTML Layouts

You can use one of the following methods to open and edit an HTML Layout:

® Open the file that contains the HTML Layout. Right-click the <OBJECT> tag for the HTML Layout, and then click **Edit HTML Layout**.

® Double-click the HTML Layout (.alx) file in the **FileView** tab.

® Open the .alx file in an ASCII editor, such as Notepad.

1. What is the main difference between ActiveX controls and Java applets?

{ewc mvimg, mvimage,!answer.bmp}

A. ActiveX controls have a visual user interface, while Java applets do not.

{ewc mvimg, mvimage,!answer.bmp}

B. Java applets are affected by the user-specified safety level, while ActiveX controls are not.

{ewc mvimg, mvimage,!answer.bmp}

C. ActiveX controls consist of compiled code that must be installed on the user's computer, while Java applets contain code that must be interpreted.

{ewc mvimg, mvimage,!answer.bmp}

D. ActiveX controls are interpreted by a Visual Basic Script (VBScript) interpreter, while Java applets are interpreted by a JavaScript interpreter.

2. Why would you use an HTML Layout?

{ewc mvimg, mvimage,!answer.bmp}

A. To place and size controls in a two-dimensional format.

{ewc mvimg, mvimage,! answer .bmp}

B. To create a template for HTML pages.

{ewc mvimg, mvimage,! answer .bmp}

C. To add Java applets to a Web page.

{ewc mvimg, mvimage,! answer .bmp}

D. To send form data to a Web server.

3. Which attribute is required in the <OBJECT> tag?

{ewc mvimg, mvimage,! answer .bmp}

A. CODEBASE

{ewc mvimg, mvimage,! answer .bmp}

B. ID

{ewc mvimg, mvimage,! answer .bmp}

C. CLASSID

{ewc

mvi
mg.
mvi
ma
ge.!
ans
wer
.bm
p}

{ew
c
mvi
mg.
mvi
ma
ge.!
ans
wer
.bm
p}

D. WIDTH

4. Which attribute(s) is required in the <APPLET> tag?

{ew
c
mvi
mg.
mvi
ma
ge.!
ans
wer
.bm
p}

A. CODE, WIDTH, and HEIGHT

{ew
c
mvi
mg.
mvi
ma
ge.!
ans
wer
.bm
p}

B. CODE

{ew
c
mvi
mg.
mvi
ma
ge.!
ans
wer
.bm
p}

C. NAME and CODE

{ew
c
mvi
mg.

D. NAME, WIDTH, and HEIGHT

{mvimage,!answer.bmp}

5. How do you specify a .cab file to be used by an ActiveX control?

{ewc mvimg,mvimage,!answer.bmp} A. ARCHIVE=file.cab

{ewc mvimg,mvimage,!answer.bmp} B. VALUE=file.cab

{ewc mvimg,mvimage,!answer.bmp} C. CODETYPE=file.cab

{ewc mvimg,mvimage,!answer.bmp} D. CODEBASE=file.cab

6. How do you specify a .cab file to download a Java applet?

{ewc mvi A. ARCHIVE=file.cab

mg.mvimage.!answer.bmp}

{ewc mvimg.mvimage.!answer.bmp}

B. <PARAM name="cabbase" value="file.cab">

{ewc mvimg.mvimage.!answer.bmp}

C. CODEBASE="file.cab"

{ewc mvimg.mvimage.!answer.bmp}

D. <PARAM name="codebase" value="file.cab">

7. What do you guarantee by signing an ActiveX control?

{ewc mvimg.mvimage.!answer.bmp}

A. The control is safe to download.

{ewc mvimg.mvi

B. The control has been written by the specified author.

{ewc mvimg, mvimage,!answer.bmp}

C. The control is safe to be used in a script.

{ewc mvimg, mvimage,!answer.bmp}

D. The author of the control is a licensed control developer.

{ewc mvimg, mvimage,!answer.bmp}

Internet Component Download is the component of Microsoft Internet Explorer that searches for ActiveX controls, and then downloads them if they are not present on the user's computer.

How Internet Component Download Works

When Microsoft Internet Explorer encounters an <OBJECT> tag, it tries to locate the control on the user's computer by first searching the registry. If the control is not registered, the browser downloads the object from the location specified in the CODEBASE attribute.

For information about the <OBJECT> tag or the CODEBASE attribute, see The <OBJECT> Tag or Setting the CODEBASE Attribute.

If the object specified by the CLASSID attribute is registered on the user's computer, Microsoft Internet Explorer tries to load (or instantiate) the object. If instantiation fails, Microsoft Internet Explorer attempts to download the code as if the object were not registered on the user's computer.

The following illustration shows a flowchart of how Microsoft Internet Explorer downloads code.

{ewc MVIMG, MVIMAGE,!W04G105.bmp}

Internet Component Download installs controls in the folder \Windows\Occache. Downloaded code will not be removed automatically when the user empties the folder of temporary Internet files.

For more information about Internet Component Download in Microsoft Internet Explorer 3.0, go to the Internet Component Download specification Web page by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

The Internet Search Path

If you do not specify a CODEBASE attribute for an object, or if the location you specify does not contain the files you want, Internet Component Download will attempt to locate the object by using the Internet Search Path. This path is a list of Object Store servers that are queried each time components are downloaded. An Object Store on the Internet Search Path is an HTTP server that processes requests for downloadable code.

The search path is specified as a string in the registry under the key HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Internet Settings\CodeBaseSearchPath.

The following syntax specifies the value for this registry key:

```
CodeBaseSearchPath = URL1; URL2; ... CODEBASE; ... URLn
```

This is a string where URL1 through URL*n* are absolute URLs that point to HTTP servers that act as Object Stores.

When locating an object, Internet Component Download will search the locations in the search path. It will use the first successful response from a server in the Internet Search Path, and will not continue searching for newer versions of components.

If the CODEBASE keyword is not included in the CodeBaseSearchPath key, Internet Component Download will not check the key for downloadable code.
{ewc mvimg, mvimage,!tip.bmp}

You can automate the process of creating and inserting a new HTML Layout by running the Template Page Wizard and selecting the Layout template. The wizard will automatically create the .htm and .alx files, add them to your Web project, and then open the .alx file in the HTML Layout Editor.

The **Command** object contains a definition of an SQL command that you want to run against a data source. You can use the **Command** object to execute any SQL statement. For example, you can execute an SQL statement that returns a recordset, updates the database, calls a stored procedure, or manipulates the structure of the database.

The collections, methods, and properties of the **Command** object vary, depending on the database provider.

To see a demonstration of how to use the **Command** object to invoke a stored procedure, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Creating a Command Object

To create a **Command** object, you pass *ADODB.Command* as an argument to the **CreateObject** function. You set properties of the **Command** object to specify the SQL statement you want to execute, the length of time you want to wait for a response to the command, and which connection to use. Finally, you call the **Execute** method to run the command.

The following example code retrieves a value from the **Request** object in an .asp file, creates a **Command** object, sets properties, and then runs the command.

```
frmClassID = Request("classID")
Set cmd = Server.CreateObject("ADODB.Command")
cmd.CommandText = "exec ClassList" & frmClassID
cmd.CommandTimeOut = 30
cmd.ActiveConnection = conn
set rs = cmd.Execute
```

The **ActiveConnection** property must be set to a valid **Connection** object variable or to a string that provides connection information.

The syntax for the **Execute** method is as follows. To see a description for each argument, click the argument.

command.**Execute** *RecordsAffected*, *Parameters*, *Options*

The following example code runs an SQL command that changes the last name of a student. The code then displays the number of records affected by the update.

```
sql= "UPDATE Students SET Last_Name= " & _
        "'" & frmNewName &"'" & _
        " WHERE StudentID = " & frmStudentid
cmd.CommandText = sql
cmd.Execute iRecordsAffected
Response.Write "Number of Records updated = " &  iRecordsAffected
```

To execute an SQL command that returns a recordset, save the return value from the **Execute** method in a recordset object variable. This example code runs the stored procedure ClassList, which returns a recordset.

```
cmd.CommandText = "Exec ClassList " & frmClassID
set rs = cmd.Execute
```

Running Stored Procedures with Parameters

To run a stored procedure that accepts parameters, you can create a **Parameter** object for each parameter, and then append the **Parameter** object to the **Parameters** collection of the **Command** object.

To create a **Parameter** object, you invoke the **CreateParameter** method of the **Command** object.

The syntax of the **CreateParameter** method is as follows. To see a description for each argument, click the argument.

command.**CreateParameter** *Name*, *Type*, *Direction*, *Size*, *Value*

To see a code sample that runs a command with parameters, click this icon

{ewc mvimg, mvimage,!code.bmp}

Note You can pass parameters by assigning values to the **Parameters** collection without using **CreateParameter**. However, the disadvantage to this method is that each assignment causes the **Command** object to query the data source for the type of the parameter.

Using **CreateParameter** to create parameters explicitly, and then appending the **Parameter** objects to the **Command** object, requires a few more lines of code but will avoid extra network trips to the database.

If a stored procedure returns an output parameter and a recordset, you must read the output parameters before accessing the recordset. Once you access the recordset, the output parameters can no longer be read.

The *RecordsAffected* argument is a Long variable to which the database provider returns the number of records affected by the SQL command that was executed.

The *Parameters* argument is an array of values used as parameters for the command.

The *Options* argument is a numeric value that indicates the type of command, such as an SQL string, a stored procedure, or a table name.

® adCmdText, 1 = Evaluate CommandText as a text definition of a command.

® adCmdTable, 2 = Evaluate CommandText as a table name.

® adCmdStoredProc, 4 = Evaluate CommandText as a stored procedure.

® adCmdUnknown, 8 = The type of command in the CommandText argument is not known.

A string representing the name of the **Parameter** object.

Optional. A Long value specifying the data type of the **Parameter** object.

Optional. A Long value specifying whether the **Parameter** object represents input, output, or both.

Optional. A Long value specifying the maximum length for the parameter value in characters or bytes.

Optional. A variant specifying the value for the **Parameter** object.

To retrieve records from a database, you create a **Recordset** object. You use properties and methods of the **Recordset** object to manipulate the data in the recordset.

To see a demonstration of how to create a recordset, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Creating a Recordset

There are three ways that you can create a recordset. You can call the:

® **CreateObject** method to create a **Recordset** object, and then use the **Open** method to retrieve records.

® **Execute** method on a **Connection** object.

® **Execute** method on a **Command** object.

In this topic, you will learn how to call the **CreateObject** method to create a recordset. For information on the other methods used to create a recordset, see Establishing a Database Connection in this chapter and Executing a Command in this chapter.

To define a **Recordset** object variable, you pass *ADODB.Recordset* as an argument to the **CreateObject** function. To create the **Recordset** object, you use the **Open** method of the **Recordset** object.
{ewc mvimg, mvimage,!tip.bmp}

The **Open** method uses the following syntax. For a brief description of each argument, click the argument name.

recordset.**Open** *Source*, *ActiveConnection*, *CursorType*, *LockType*, *Options*

The following example, uses the **CreateObject** function to define a recordset object variable and then uses the **Open** method to create the **Recordset** object. The example passes a connection string when invoking the **Open** method of the **Recordset** object.

```
Const adOpenKeyset = 1
Const adLockOptimistic = 3

set rs = Server.CreateObject ("ADODB.Recordset")
rs.Open "Select * from students", "DSN=StateU;UID=SA;PWD", _
    adOpenKeyset, adLockOptimistic
```

When you pass a connection string to the **Open** method of the **Recordset** object, a connection will be created for you. However, a separate connection is opened for each recordset you create using this method.

If you want to share a connection with multiple recordsets, create the **Connection** object first, and then pass the **Connection** object variable, rather than a literal string, when invoking the **Open** method of a **Recordset** object as shown in this example.

```
set conn = Server.CreateObject ("ADODB.Connection")
conn.Open "DSN=StateU;DATABASE=StateU", "SA", ""

set rs = Server.CreateObject ("ADODB.Recordset")
rs.Open "Select * from students", conn, _
        adOpenKeyset, adLockOptimistic
```

Cursor Types

When you open a recordset, it contains a cursor. A cursor points to the current record. When you call the **MoveNext** or other navigation methods, you move the cursor. When you create a recordset, you can specify which type of cursor to create. Each cursor type supports different functions. To determine the cursor type you want to create, you need to determine the features you need and which cursor types are included by the database provider you are using.

The following paragraphs describe the features of the different cursor types. For more information about the different types of cursors, see the article #Q149054, #Q149054 Choosing a rdoResultset CursorType in the

Knowledge Base section of the Library.

Forward Only

A forward-only cursor is the fastest and most efficient type of cursor. You can use it only to move forward through a recordset, and it is read-only. Forward-only cursors are useful when you want to read once through the recordset and display the data.

Keyset

A keyset cursor is relatively fast because it assigns a key value to each record and stores only the keys. You can use it to move forward and back through a recordset and update the data. When you use a keyset cursor, you will see modifications and deletions by other users, but you will not see any new records added by other users. Keyset cursors are useful when you want to move forward and back through a cursor and update the data.

Dynamic

The dynamic cursor provides the most up-to-date view of your data and requires the most overhead. When you use a dynamic cursor, you will see any modifications and inserts made by other users to the data. This cursor is useful if you require the most current view of the data.

Static

A static cursor enables you to move forward and back through a recordset, but is read-only and does not reflect any updates by other users. This cursor type is useful when you want to move in either direction, but do not need to update the data.

Optional argument. A **Command** object variable name, an SQL statement, a table name, or a stored procedure call.

Optional argument. A **Connection** object or a string that contains connection information.

A value that indicates the type of cursor to create.

- 0 = forward-only cursor
- 1 = keyset cursor
- 2 = dynamic cursor
- 3 = static cursor

A value that indicates what type of locking should be used.

- 0 = Read-only
- 1 = Pessimistic
- 2 = Optimistic
- 3 = Batch Optimistic

A numeric value that indicates how the Source argument should be interpreted if it is not a Connection object.

- 1 = Evaluate Source as a text definition of a command, such as an SQL statement.
- 2 = Evaluate Source as a table name.
- 4 = Evaluate Source as a stored procedure.
- 8 = The type of command in the Source argument is not known.

In this topic, you will learn about strategies for handling database errors in a Web page.

When you create Web applications that access a database, you should anticipate possible database errors and include error-handling code in your script. For general information on how to write error-handling code in VBScript, see Handling Run-Time Errors in Chapter 5: Adding Client-Side Script.

To see a demonstration of how to handle errors, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Error Handling Strategy

The best strategy for handling errors is to provide code that attempts to correct the error. For example, if opening a database connection fails, you can write code to try connecting to a back-up database. If this works, the user doesn't need to know that an error has occurred.

You can also try to prevent errors. For example, if you have a form in which the user enters a date range, you can place validation code on the form to verify the dates before the form is submitted.

If an error cannot be corrected or prevented, you can return an informative message to the user. One way to supply a message is to programmatically switch to another Web page that will display the message.

To programmatically switch to another Web page, you call the **Redirect** method of the **Response** object. However, **Redirect** will work only if it has been placed in the server script before the <HTML> tag is read. By placing all of the server script before the <HTML> tag, you can trap errors and display a different Web page if necessary. For more information on using **Redirect**, see Using the Redirect Method in Chapter 6: Using Active Server Pages.

To see a code sample that checks for a database error and programmatically switches to a Web page that displays an error message, click this icon.
{ewc mvimg, mvimage,!code.bmp}

The Errors Collection

The **Connection** object provides an **Errors** collection which contains information on database errors. To determine if an error occurred, you can use the **Err** object provided by VBScript or the **Errors** collection.

The advantage of using the **Errors** collection is that if multiple errors occur during a single database operation, all of the errors will be stored in the collection. The **Err** object contains information only on the last error returned.

In ADO, you create objects independently, which enables you to write simpler code and create only the objects you need for a specific task. In DAO, you create one object to get to another.

The following illustration shows the relationship among the ADO objects.

{ewc MVIMG, MVIMAGE,!W07G010.bmp}

Note There are situations in which ADO does require you to create one object to get to another. In these situations, ADO will create objects implicitly.

In ADO, you work primarily with the objects **Connection**, **Command**, and **Recordset**. Both the **Connection** object and the **Command** object provide an **Execute** method that runs a command and can return a recordset. You can also use the **CreateObject** function to create a recordset object variable explicitly, and then use the **Open** method of the object variable to create the recordset.

Connection

The **Connection** object represents an open connection to a data source or OLE DB provider. You use the **Connection** object to cache an open connection for use by other objects, such as the **Recordset** object.

The **Connection** object has an **Errors** collection, which stores any errors that occur when accessing the data.

Recordset

The **Recordset** object represents an entire set of records from a query result. You use a **Recordset** object to retrieve, display, and update data in a database.

Command

The **Command** object stores the definition of a specific command that you intend to run with a data source. You can use the **Command** object to run any SQL command. For example, you can run stored procedures, insert or update data, or change the structure of a database.

Note If the OLE DB provider or ODBC driver does not support running commands, you will not be able to create a **Command** object.

A **Connection** object represents an open connection to a data source or OLE DB provider. You can use the **Connection** to run commands or queries on the data source. When a recordset is retrieved from the database, it is stored in a **Recordset** object.

To see a demonstration of how to establish a connection and retrieve a recordset, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Opening a Connection

To establish a connection with a data source, you first create a **Connection** object. To create a **Connection** object, you call **CreateObject** and pass the parameter ADODB.Connection.

To define the connection, you set properties for the **Connection** object. Typically, the two properties you will set are **ConnectionTimeOut** and **CommandTimeOut**, which are both measured in seconds. **ConnectionTimeOut** determines how long the object will wait before timing out when connecting to a data source. **CommandTimeOut** determines how long the object will wait for the results of a command or query.
{ewc mvimg, mvimage,!tip.bmp}

To connect with the data source, you use the **Open** method The syntax for the **Open** method is as follows. To see a description of each argument, click the argument.

connection.**Open** *ConnectionString*, *User*, *Password*

After the **Open** method succeeds in connecting with the data source, you can run queries.

The following example code creates a **Connection** object, opens a connection to the StateU data source, and runs a query.

```
Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionTimeOut = 10
conn.CommandTimeOut = 20
conn.Open "DSN=StateU;DATABASE=StateU", "SA", ""
set rs = conn.Execute ("Select * from students")
```

If you are connecting to an OLE DB database, set the **Provider** property of the **Connection** object to the OLE DB provider name, or put the OLE DB provider name in the connection string. For more information on using OLE DB providers, see the article #Q164586, How to Use OLE DB Sample Text Provider in ADO in the Knowledge Base section of the Library.

Connection Object Methods

The **Connection** object provides an **Execute** method that runs an SQL query. You can also use the **BeginTrans**, **CommitTrans**, and **RollbackTrans** methods to create transactions. For more information on using transactions, see Chapter 9: Using Microsoft Transaction Server.

Using Connection Information

Rather than using literal values for the **Connection** object properties, you can use session variables. When you use Visual InterDev to add a data connection to a Web project, Visual InterDev adds script to the Global.asa file that stores information about the connection in session variables. You can access these session variables from code in an .asp file.

The advantage to using session variables when creating a connection is that you can change the value of the variables in the Global.asa file, and the new values will be used by any of your .asp files that refer to the variables. If you change the properties of a data connection in Visual InterDev, it modifies the script in the Global.asa file to set the variables to the new values.

The following example code shows how you can use session variables to create a connection.

```
Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionTimeout = Session("StateU_ConnectionTimeout")
conn.CommandTimeout = Session("StateU_CommandTimeout")
conn.Open Session("StateU_ConnectionString"), _
```

```
        Session("StateU_RuntimeUserName"), _
        Session("StateU_RuntimePassword")
```

Closing a Connection

When you have finished working with the database, you use the **Close** method of the **Connection** object to free any associated system resources. Using the **Close** method does not remove the object from memory. You can change the object's properties, and then use the **Open** method to open it again. To completely remove an object from memory, you set the object variable to **Nothing**.

This example code closes a data connection and sets the object variable to **Nothing**.

```
conn.Close
Set conn = Nothing
```

Note If you do not close a **Connection** object, it will close automatically when the .asp file has finished running. If you want to keep a connection open during a session or application, you can store the **Connection** object as a variable in the **Session** or **Application** objects.

When to Use a Connection Object

An open connection is an expensive resource because the data source may only allow a limited number of connections. The more open connections there are at one time, the more likely it will be that another application will be denied access to the same data source. It is therefore important not to have a connection open any longer than necessary.

However, the server activity required to open a connection also consumes multiple CPU cycles. Opening and closing connections repeatedly can be costly on a server in high demand. Therefore, if a specific connection will be used again relatively soon, it is a good idea not to close the connection.

Note You can have Microsoft Internet Information Server (IIS) optimize data source connections for you by turning on connection pooling. Connection pooling creates a pool of connections that can be used by applications. When an application requests a connection, it can be retrieved from the pool. If a connection has not been used for a certain amount of time, IIS will eventually drop the connection from the pool.

To enable connection pooling, set the StartConnectionPool registry entry to 1 (True). The following text shows the location of this registry entry.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\ASP\
    Parameters\StartConnectionPool
```

Information, such as the data source name, used for establishing a connection to a data source. The information required for a connection string is defined by the OLE DB provider.

A string that contains the user name for establishing a connection.

A string that contains the password for establishing a connection.

After you have defined the Web page, you insert the **AdvancedDataControl** object and set the appropriate attributes.

The **AdvancedDataControl** object runs queries and makes the resulting recordset available to data-bound controls on the Web page. You bind data from the **AdvancedDataControl** to data-aware controls by listing the data-aware controls in the **Bindings** attribute of the **AdvancedDataControl**.

To see a demonstration of how to create a Web page that uses the **AdvancedDataControl** object, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Adding the AdvancedDataControl Object

To add the AdvancedDataControl object

1. In the Visual InterDev editor, open the Web page.
2. Right-click in the <Body> section of the Web page.
3. Click **Insert ActiveX Control**, and then click **AdvancedDataControl**.
4. Set the ID attribute to name the control.
5. Set the CODEBASE attribute to **msadc10.cab**, which is the installation file for the Advanced Data Connector client components.

Note Each of the Advanced Data Connector components have two CLSIDs in the registry. This causes two entries to display for each control in the **Insert ActiveX Control** dialog box.

Either of the two entries will insert the same ADC control. In future releases of the ADC, each entry will support different features.

Setting Attributes

The following table describes the attributes you set for the **AdvancedDataControl** object.

Attribute	Description
SQL	Specifies the SQL statement to retrieve records.
BINDINGS	Specifies the IDs of the data-bound controls that are bound to the **AdvancedDataControl** object. To specify multiple data-bound controls, separate each ID with a semicolon (**;**).
CONNECT	Specifies the data source name, user ID, and password.
SERVER	If you are using HTTP, ServerName is the name of the Web server computer.

The following HTML shows the HTML <OBJECT> tag that inserts the **AdvancedDataControl**.

```
<!--Grid Data Bound Control -->
<OBJECT ID="GRID" WIDTH=700 HEIGHT=200
    CLASSID="CLSID:BC496AE0-9B4E-11CE-A6D5-0000C0BE9395"
    CODEBASE="HTTP://myserver/MSADC/samples/Sheridan.cab"
</OBJECT>
<!--AdvancedDataControl object -->
<OBJECT CLASSID="clsid:9381D8F2-0288-11d0-9501-00AA00B911A5"
    ID=ADC
    CODEBASE="//myserver/MSADC/msadc10.cab">
    <PARAM NAME="BINDINGS" VALUE="Grid;">
    <PARAM NAME="SQL" VALUE="select * from classes">
    <PARAM NAME="CONNECT" VALUE="DSN=StateU;UID=sa;PWD=;">
    <PARAM NAME="SERVER" VALUE="http://myserver">
```

```
</OBJECT>
```

Note You must add the data-bound Grid control before the **AdvancedDataControl** object on the HTML page. For more information, see the README file installed with the Advanced Data Connector.

You can add client-side script to your Web page to change properties of the **AdvancedDataControl** object and submit a new query to the Web server at run time. The only property that must be set at design time is BINDINGS.

Sending a New Query

To send a new query to the Advanced Data Connector, you modify the **SQL** property of the **AdvancedDataControl** object and use the **Refresh** method to run the query. A new set of records will be retrieved and the data-bound controls will be updated automatically with the new data.

The following example code sets the SQL property of the **AdvancedDataControl** object based on input from a user, then queries the database again.

```
Sub cmdFind_OnClick
    ADC.SQL = "exec classlist " & "'" & txtid.value & "'"
    ADC.Refresh
End Sub
```

Changing the Current Record

The **AdvancedDataControl** object uses a recordset, which always specifies a current record. The current record is displayed in the data-bound controls.

To change the current record, you run script that uses the **Move** methods of the **AdvancedDataControl** object. The data-bound controls will display the new current record. Because the recordset is cached on the client workstation, a user can browse a large recordset without sending additional requests to the Web server for new Web pages.

The following example code moves the current record forward one record.

```
Sub cmdMoveNext_OnClick
    ADC.MoveNext
End Sub
```

Changing Data

Data-bound controls enable the user to visually edit, add, or delete records. All changes made by the user are stored locally until the user explicitly submits or cancels the update.
{ewc mvimg, mvimage,!tip.bmp}

Note To enable users to change data on data-bound controls, you add the **For Browse** statement to the end of the query. Currently, only Microsoft SQL Server supports this statement.

For information about the **For Browse** statement, see the Readme file installed with the Advanced Data Connector.

To submit changes, invoke the **SubmitChanges** method of the **AdvancedDataControl** object. To cancel changes, invoke the **CancelUpdate** method.

The following example code shows how an **Update** button and a **Cancel** button can be used to submit or cancel changes to a recordset.

```
Sub Update_OnClick
    ADC.SubmitChanges
End Sub

Sub Cancel_OnClick
    ADC.CancelUpdate
End Sub
```

There are three steps to building a Web application that uses the Advanced Data Connector.

1. Define the HTML page.
2. Insert the **AdvancedDataControl** object on the Web page.
3. Set properties of the **AdvancedDataControl** to submit a query. Most properties can be set at design time or with client-side script.

To define your Web page, add text, command buttons, and data-bound ActiveX controls. The following table lists the data-aware controls that have been tested and work with the **AdvancedDataControl** object and associated client-side components of the ADC.

Control Name	File name	Source
SSDBCombo or SSDBGrid	SSDATB32.ocx	Visual Basic version 4.0 or Sheridan Software
DBListBox or DBComboBox	DBLIST32.ocx	Visual Basic version 4.0
True DBGrid	TDBGS32.ocx	Apex Software
MhImage	MHIMAG32.ocx	MicroHelp Software
Rich Text Field	Richtx32.ocx	Visual Basic 4.0

The following illustration shows the HTML used to define a Web page on which a user can enter a class ID to retrieve a list of students in the class. To see the HTML code that defines input fields to enter the class ID, command buttons to run the query, a grid to display the results, and command buttons to move to the next and previous records in the recordset, click the illustration.

{ewc MVIMG, MVIMAGE,!w07g020.shg}

```
<BODY>
<!-- Field in which user enters class ID -->
Enter Class ID
    <INPUT SIZE=30 NAME="txtID">
    <INPUT TYPE=BUTTON VALUE="Find" NAME="cmdFind">
<p>
<!--databound grid control -->
    <OBJECT ID="Grid" WIDTH=400 HEIGHT=200
     CLASSID="CLSID:BC496AE0-9B4E-11CE-A6D5-0000C0BE9395"
     CODEBASE="HTTP://myserver/MSADC/samples/sheridan.cab">
        <PARAM NAME="Caption" VALUE="Students">
    </OBJECT>
<p>
<p>
    <INPUT TYPE=BUTTON VALUE="Next" NAME="cmdMoveNext">
    <INPUT TYPE=BUTTON VALUE="Previous" NAME="cmdMovePrevious">
</BODY>
```

The components of the Advanced Data Connector are divided into client-side components that run on a Web browser and server-side components that run on a Web server.

To see an animation of how the Advance Data Connector is used to retrieve and update records from a database, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

Client Components

ADC client components are ActiveX controls that run on a Web page to provide dynamic data to the user. These controls are hidden on the Web page. The controls display data from a recordset in data-bound ActiveX controls that are visible. This list describes the ADC client components.

® **AdvancedDataControl** Object

The **AdvancedDataControl** object runs queries and makes the resulting recordsets available to the data-bound controls on a Web page. You set properties for the object to identify the Web server, data source, and SQL statement to retrieve records.

You use the <PARAM> tag to set the BINDINGS property to indicate which data-bound controls to use to display the data to the user.

® **AdvancedDataSpace** Object

The **AdvancedDataSpace** object creates instances of business objects that have been implemented as ActiveX Server Components and reside on a Web server.

You can write client-side script to use this object to invoke instances of your own custom business objects on the Web server.

® **ADOR.Recordset** Object

The **AdvancedDataSpace** object creates an **ADOR.Recordset** object when it retrieves records. This type of recordset object is similar to the ADO **Recordset** object, but does not include all of the features of an ADO **Recordset** object. Because it includes fewer features, it is smaller and can download to the client quickly.

Server Components

The server-side components of the Advanced Data Connector include the **AdvancedDataFactory** Active Server Component and the Advanced Data Connector application.

® **AdvancedDataFactory**

The **AdvancedDataFactory** object is a business object that has been implemented as an ActiveX Server Component. This is the default object used by the **AdvancedDataControl** to runs queries.

® Advanced Data Connector Application

The Advanced Data Connector ISAPI (ADISAPI) is an ISAPI DLL that runs on a Web server and communicates between the client workstation and the business objects.

How ADC Displays a Recordset

When a user submits a query on a Web page, the client-side script assigns the query to the **AdvancedDataControl** object and calls the **Refresh** method. The **AdvancedDataControl** object then submits the query by using HTTP to the Web server.

The ADISAPI application routes the query to the **AdvancedDataFactory** object, which runs the query against the data source.

The resulting recordset is sent back to the **AdvancedDataControl** object by using HTTP. The data-bound controls on the Web page display records from the recordset.

The recordset is cached on the client side. When a user moves through the recordset, the controls will display the data without having to make another trip to the Web server.

In this section, you will learn how to use Visual InterDev and Visual SourceSafe to create a new Web project, and how to add HTML files to a Web page.

The user interface for Visual InterDev helps you quickly create and organize Web projects. A Web project contains a set of files and objects that are combined to make up a Web site.

Visual InterDev enables the members of a Web development team to retrieve files from the Web server and work on them locally. Visual SourceSafe provides additional file management features for the development team that includes checking files in and out of a project.

This section includes the following topics:

With Visual InterDev, you work with files on a development computer, as well as with files on other Web servers. The various members of the Web development team can access files and work on them simultaneously in a managed environment.

To work with a file in Visual InterDev, you access it, and then copy it to your local computer. When you have finished working on a file, you release the file back to the Web server so that other team members can view the changes or make their own changes.

To see a demonstration of how to create a new Web project and work with Visual InterDev files, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Visual InterDev Projects

When you create a Web site in Visual InterDev, the files that make up your Web site are stored together on a Web server. To access Web files from the Web server, you must first create a local project file that points to the Web server and the specific Web site.

The local project file has an extension of .dsp. To work with the files in a project, you must open the project file from within Visual InterDev.

Web files on the server, together with the local project file, make up a Web project.

To create a new Web project

1. On the **File** menu, click **New**.
2. Click the **Projects** tab.
3. Select the **Web Project Wizard**.
4. Type the name of the local project file, and then click OK.
5. Type the name of the Web server for the project, and then click **Finish**.

   You can type the name of a new Web server or select an existing Web server.

Visual InterDev creates a new Web project on your computer and displays it in FileView in the Workspace window.

By default, Visual InterDev creates an \Images folder and a Global.asa file in the project. If you want to add search capabilities to your Web site, Visual InterDev will create the file search.htm.
{ewc mvimg, mvimage,!tip.bmp}

To see an illustration of a Visual InterDev workspace that contains a new project, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Visual InterDev Workspaces

The Visual InterDev workspace is where you do all of your work. A workspace can contain one or more projects, and you can work on different projects at the same time without having to open and close them.

When you create a new Web project, you have the option of either adding the project to an open workspace or creating a new workspace. You can have only one current workspace.

When you open a new workspace, any currently open workspaces and their projects will be closed. Therefore, to work on more than one project at a time, add the additional projects to the current workspace.

The projects in a workspace can be of the same or different types. For example, a workspace can include a Web project and a database project, or two Web projects, each pointing to different Web sites.

A workspace keeps track of files on the local computer and on the Web server. It also keeps track of the files that are currently open.

When you exit Visual InterDev, it saves your workspace as a file with a .dsw extension. When you restart Visual InterDev, you can return to the project and files you were working on by selecting the workspace from the **Recent Workspaces** list on the **File** menu.

With Visual InterDev, you can easily create, copy, delete, and rename files in a Web project.

To see a demonstration of how to add and delete files in a Web project, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Updating Project Status

As you and other team members add and delete files from the Visual InterDev project, files are added and deleted on the Web server but not on your local computer.

To see the most recent files in a project, right-click the project name in FileView, and then click **Refresh**. This synchronizes your local view of the project with the files included on the Web server.

Creating New Files

To create a new file in a Visual InterDev project, you use the following procedure.

u To create a new file in a project

1. Click **New** on the **File** menu.
2. On the **Files** tab of the **New** dialog box:
   a. Select the type of file you want to add.
   b. Select the **Add to project** option.
   c. Select the target project from the list of open projects.

      The default project is the current project.
   d. Type a file name for the new file.
   e. Verify the file location, or type the full path for the new file, and then click OK. The location of the file must be within the working folder.

      If you want to add the file to a Web project, a master copy will be added to the Web folder, and will be displayed in FileView with the appropriate icon for the type of file created.

Adding Existing Files

If you have existing files you want to add to your Web project, you can select the files in Windows Explorer, drag them to the **FileView** tab of the Visual InterDev workspace, and then drop them into the project. Or, you can right-click the project folder, and then click **Add File**.

u To add an existing file to a project

1. In the project workspace, click the **FileView** tab.
2. In the project, right-click the folder in which you want to insert the file, and then click **Add File**.

   To add all of the files in a folder, click **Add Folder Contents**.
3. Select the file or files you want to add, and then click OK.

   The files will be copied to the Web project folder and stored on the Web server. An icon for each file will be displayed in FileView.

Deleting Files

If you want to delete files from a Web project, the files will be removed from your local computer and from the Web server.

u To delete files or folders

1. In the project workspace, click the **FileView** tab.
2. Select the files or folders you want to delete.
3. Click **Delete** on the **Edit** menu.

Renaming Files

You can also rename files in your Web project. Visual InterDev will automatically fix any hyperlinks from other files to the new file name. However, before you can rename a file, you must close any local copies of the file.

u To rename files or folders

1. In the project workspace, click the **FileView** tab.
2. Right-click the file you want to rename, and then click **Rename**.
3. Type a new name for the file.

   Visual InterDev displays a dialog box, which asks if you want to have the hyperlinks fixed for you.

   To see an illustration of the dialog box, click this icon.
   {ewc mvimg, mvimage,!illust.bmp}

With Visual InterDev, multiple members of a Web development team can change local copies of Web files at the same time. However, this feature can create a problem if the local file is sent back to the server so that one team member overwrites changes made to another local copy of the file.

To prevent team members from overwriting each other's work, you can install a source code control system on a Web server, such as Microsoft Visual SourceSafe. Visual SourceSafe is a project management tool that enables users to check files in and out of a server.

To customize the way in which Visual InterDev works with Visual SourceSafe, you click **Options** on the **Tools** menu, and then click the **Source Control** tab.

To see an illustration of the **Source Control** tab, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

To see a demonstration of how to use Visual SourceSafe with Visual InterDev, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

For more information about using Visual SourceSafe with Visual InterDev, search for "Developing applications in teams" in the Visual InterDev InfoViewer.

How Visual SourceSafe Works

After you install Visual SourceSafe on a Web server and enable the source control system for a Web site, you can open files just as you did without Visual SourceSafe. When you want to check out a working copy of a file, and the file has not been checked out by someone else, Visual SourceSafe provides you with a local, write-enabled copy of the file.

After you have a write-enabled copy of the file, Visual SourceSafe marks the file as checked out so no one else can edit it. When you check in the working copy, Visual SourceSafe marks the file as checked in so someone else can check out and edit the file.

If you request a working copy of a file that another user has already checked out, Visual SourceSafe will display a message. If you still want the file, it provides you with a read-only version. Multiple users can have read-only copies of the file, but only one user can have a write-enabled copy.

To see an illustration of the menu options for Visual SourceSafe, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

In Visual InterDev, you can use Link View to see a visual representation of the hyperlinks and source files on any Web page. Link View shows the various elements (such as other Web pages and objects) that are linked to by a Web page or set of Web pages.

The following illustration shows the Link View window for the State University Web site.

{ewc MVIMG, MVIMAGE,!W02g125.bmp}

You can use Link View to see the links of your own Web site or on any Internet or intranet site.

To view the links of a Web page in your own Web site, right-click a file in the Web project, and then click **View Links**. To view the links of a Web page in another Web site, click **View Links on WWW** on the **Tools** menu, and then type the URL of the Web site.

Verifying Objects

When you open a Web page in Link View, Visual InterDev reads the HTML code, locates the elements used to build the Web page, and provides graphical information about whether the objects are valid or not.

Filtering Views

To display only specific types of objects in Link View, you can add a filter by clicking the appropriate button on the **Link View** toolbar. For example, to display only HTML pages in Link View, click the **Show HTML Pages** button.

The following illustration shows the buttons on the **Link View** toolbar.

{ewc MVIMG, MVIMAGE,!W02G135.bmp}

The HTML Author is responsible for creating a cohesive presentation of the content in a Web site.

Responsibilities

The HTML Author has the following responsibilities:

® Create HTML pages

The HTML Author creates and maintains HTML pages by adding appropriate HTML tags.

® Create hyperlinks

The HTML Author ensures that Web site navigation is easy and complete by adding hyperlinks.

Tools

The tools that enable HTML Authors to complete their responsibilities include:

® Microsoft FrontPage 97

The HTML Author uses FrontPage 97 to create and edit HTML pages.

To see an illustration of FrontPage 97, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

® Microsoft Office 97

The applications in Microsoft Office 97 enable the HTML Author to create a wide range of HTML pages. You can use any of the wizards in these applications to create Web pages. Or, on the **File** menu of any Microsoft Office 97 application, you can click **Save As HTML**.

To see an illustration of Microsoft Word 97, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

The HTML Author for State University

As the HTML Author for the State University Web site, you will create a variety of HTML files. The following table describes these files.

File	Description
default.htm	The State University home page. This page contains three frames that reference home.htm, crest.htm, and links.htm.
home.htm	The main informational page for State University.
links.htm	A navigation page containing hyperlinks to most of the pages in the State University Web site.
crest.htm	A page that contains the State University logo.
feedback.htm	A page that gathers feedback from a student and submits it to the getfeedback.asp Active Server Page.

In this section, you will learn how to use the FrontPage Editor with Visual InterDev to create text, graphics, and hyperlinks for a static HTML page. A static Web page displays information without enabling user interaction.

There are two editors in Visual InterDev that you can use to create a static HTML page: the FrontPage Editor and the Visual InterDev Source Editor. The FrontPage Editor is a WYSIWYG editor, which is well suited for creating static HTML pages containing text and graphics.

To create a new HTML page with Visual InterDev, click **New** on the **File** menu, click the **Files** tab, and then click **HTML Page**. Visual InterDev displays an HTML page with the required HTML tags.

This section includes the following topics:

Design-time ActiveX controls are tools you can use to insert HTML tags, client-side script, or server-side script into a Web page.

Using Design-Time Controls

You insert design-time controls in the same way that you insert other ActiveX controls. Click **Insert ActiveX Control** on the **Insert** menu, or right-click the file, and then click **Insert ActiveX Control**.

You set properties for design-time controls by using the Visual InterDev Object Editor. When you close the Object Editor, Visual InterDev writes an <OBJECT> tag to the file. The <OBJECT> tag saves information for the control, and is contained in an HTML comment tag (<!-- -->) so it will not be visible to the user.

A design-time control also writes text into your Web page. The generated text can include HTML tags, server-side script, and client-side script.

The following table lists and describes the design-time ActiveX controls that ship with Microsoft Visual InterDev.

Design-time ActiveX control	Description
Data Command	Writes server-side script that executes a command, such as a query against a database.
Data Range Header	Writes server-side script that creates an ADO **Recordset** object by using a query or stored procedure. Starts a loop through the records in the recordset, and provides the option to display one record at a time.
Data Range Footer	Used in conjunction with a **Data Range Header** control, writes the server-side script that finishes the loop through all records in a recordset.
Include	Enables the contents of a file to be added to an Active Server Page before that page is processed.

To see a demonstration of how to use the **Include** design-time control, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

For information about using include files, see the technical article, Server-Side Includes in the Articles and White Papers section of the Library.

You can also create your own design-time controls by using the Microsoft Design-Time Control Software Development Kit. For information about this SDK, go to the Design-Time Control SDK Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

HTML Tags Added by Design-Time Controls

Design-time controls insert tags into the Web page. These tags define the control for future editing, and create HTML tags that are processed by the server when the user requests the HTML page.

When a design-time control adds text to an .asp file, the control-generated text is inserted between two METADATA comments: a startspan comment and an endspan comment. The <OBJECT> tag for the design-time control is contained in the first METADATA comment.

The following example code shows the METADATA startspan comment that is inserted by the **Data Command** design-time control:

```
<!--METADATA TYPE="DesignerControl" startspan
<OBJECT ID="rsTemp" WIDTH=151 HEIGHT=24
CLASSID="CLSID:7FAEED80-9D58-11CF-8F68-00AA006D27C2">
  <PARAM NAME="_Version" VALUE="65536">
  <PARAM NAME="_Version" VALUE="65536">
  <PARAM NAME="_ExtentX" VALUE="3986">
```

```
  <PARAM NAME="_ExtentY" VALUE="635">
  <PARAM NAME="_StockProps" VALUE="0">
  <PARAM NAME="DataConnection" VALUE="DBConn">
  <PARAM NAME="CommandText" VALUE="SELECT AuthorID, AuthorName FROM authors">
</OBJECT> -->
```

You can reopen the control to change its property settings by right-clicking anywhere between the startspan and endspan comments, and then clicking **Edit Design Control**.

The following example code shows where the control-generated text of the design-time control ends:

```
<!--METADATA TYPE="DesignerControl" endspan-->
```

When the .asp file is processed on the Web server, the server-side script is read and processed. The METADATA comments are stripped from the file because they are needed only by Visual InterDev. In standard HTML pages, the METADATA comments will be invisible because they are embedded in comment tags.

You can edit the script created by the design-time control. However, if you reopen the control, any changes will be overwritten. If you want to edit or add to the control-generated script, delete the METADATA comments.

Web pages that use frames include two main elements:

® **Main frame HTML file**

This file contains the tags necessary to implement each frame on a page, along with references to the HTML files for each frame. The file does not contain a <BODY> tag.

® **Source HTML files**

Each frame on a page contains its own source HTML file.

Creating Frames

u **To create an HTML page with frames**

1. Create one source HTML file for each frame on a Web page.

   The source files can contain any HTML tags.
2. Create a new HTML file that contains <HTML> and <HEAD> tags, but not a <BODY> tag.

   This is the main frame file that users open with a Web browser.
3. In the area of the document that typically contains the <BODY> tag, add <FRAMESET> tags.
4. For each frame on a page, add a <FRAME> tag, and set the SRC (source) attribute to the name of the HTML file that you want to appear in the frame.

The following example code creates two vertical frames:

```
<FRAMESET COLS="*, 2*">
    <FRAME SRC="Cell_1.htm">
    <FRAME SRC="Cell_2.htm">
</FRAMESET>
```

The left frame will be half as wide as the right frame.

The following illustration shows what you will see in the browser if you use these <FRAMESET> tags.

{ewc MVIMG, MVIMAGE,!W02g230.bmp}

The <FRAMESET> Tag

The <FRAMESET> tag defines the location, size, and orientation of frames on an HTML page. This tag has two attributes: ROWS and COLS. You can either create a frameset with rows or a frameset with columns.

The following example code defines a page with two horizontal frames:

```
<HTML>
<FRAMESET ROWS="100, *">

</FRAMESET>
</HTML>
```

The ROWS attribute defines horizontal frames. It is followed by a comma-delimited list of the sizes for each frame on the page. You can specify actual pixel sizes, percentages, or relative sizes.

In the following example code, the first frame is 120 pixels, the third frame is 20% of the total height, and the second frame occupies the remainder of the height:

```
<FRAMESET ROWS="120, *, 20%">
```

You can create vertical frames by using the COLS attribute. You specify the frame in the same way as the ROWS attribute.

This example code creates two vertical frames:

```
<FRAMESET COLS="2*, *">
```

The left frame will be twice as wide as the right frame.

For more information about the <FRAMESET> and <FRAME> tags, see the technical article, Frame Tags in the Articles and White Papers section of the Library.

Browsers That Do Not Support Frames

Not all browsers support the frames feature in HTML 3.0. As a consideration to users of these browsers, you can supply alternate HTML by placing it in the <NOFRAMES> tag of the main frame HTML file.

The <NOFRAMES> tag appears after the <FRAMESET> tag, as shown in this example code:

```
<HTML>
<FRAMESET COLS="*, 2*">
    <FRAME SRC="Cell_1.htm">
    <FRAME SRC="Cell_2.htm">
</FRAMESET>

<NOFRAMES>
<BODY>
...
</BODY>
</NOFRAMES>
</HTML>
```

With the FrontPage Editor, you can easily add text and graphics to an HTML page.

To set formatting options for a page, you use the **Format** toolbar. To see a description of some of the buttons on the FrontPage **Standard** and **Format** toolbars, click the buttons in the following illustration.

{ewc MVIMG, MVIMAGE,!W02g440.shg}

To see a demonstration of how you can add text and images to an HTML page by using the FrontPage Editor click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Text

You use the FrontPage Editor to apply formatting to an HTML page in the same way that you apply formatting to a Microsoft Office document. For example, you can cut and paste text or use drag-and-drop to move text to a new location.

To change font and paragraph properties, you use the **Format** toolbar or the **Format** menu.

Some of the available formatting options are:

® Centering the style of a paragraph to make it centered or aligned right.

® Changing the text font to make it bold, italic, or underlined.

® Changing text to a bulleted or numbered list.

Images

Images make a Web site look interesting, but many users turn off image display to download Web pages faster. Therefore, you should always provide alternate text for images on your Web pages.

The FrontPage Editor includes a library of clip art for commonly used buttons, icons, and backgrounds.

u **To insert an image with the FrontPage Editor**

1. Click the page where you want to insert the image.
2. On the **Insert** menu, click **Image**.
3. Click the **Other Location** tab, type the relative or absolute URL where the image is located, and then click OK.
4. To set properties for the image, such as alternate text, the hyperlink location, or alignment and size, right-click the image, and then click **Image Properties**.

There are two different formats in which an HTML page will be displayed:

® **Graphics Interchange Format (.gif)**

A .gif file is an encoded and compressed file for images of up to 8 bits of color.

® **Joint Photographic Expert Group (.jpeg**)

A .jpeg file is an encoded and compressed file for images of up to 24 bits of color.

If you insert a color image that is not a .gif or .jpeg file, the FrontPage Editor will automatically convert the file to one of these formats, depending on the size of the image.

This set of buttons enables you to select a font and format text.

This set of buttons indents increases or decreases the indent of a text selection.

These buttons create a numbered or bulleted list.

These buttons align text to the right, left, or center of the page.

This drop-down list box enables you to choose a style for a text selection. For example, **Heading 1**, **Definition**, and **Numbered List** are some of the styles you can select.

This button inserts a WebBot component on a page.

This button inserts a table.

This button inserts an image.

This button inserts a hyperlink.

Standard HTML controls typically reside on forms on an HTML page. On a form, controls are also known as form fields.

The following illustration shows an HTML form that contains standard HTML controls. To view each type of control on the form, click the control.

{ewc MVIMG, MVIMAGE,!W02G010.shg}

Note Microsoft Internet Explorer does not require standard HTML controls to be contained on forms, but other browsers do. Therefore, you should always place standard HTML controls on forms.

Inserting Controls

With the FrontPage Editor, you use the **Form Field** command on the **Insert** menu to select the controls you want to insert. The Visual InterDev Source Editor does not have a built-in interface for inserting HTML controls, so you must add HTML tags manually.

The following illustration shows how you can select controls with the FrontPage Editor by using the **Form Field** command.

{ewc MVIMG, MVIMAGE,!W02G030.bmp}

To set properties for each control, right-click the control, and then click **Form Field Properties**. A dialog box for that control will be displayed for you to set its properties.

To see an illustration of the **Push Button Properties** dialog box, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

The following table describes which menu item to select and which properties to set for each control.

To create this type of control	Select this menu item	And set this property
Text Box	One-Line Text Box	
Password Text Box	One-Line Text Box	Password to Yes
Multiple-line Text Box	Scrolling Text Box	
Check Box	Check Box	
Radio Button	Radio Button	
Single-select List Box	Drop-Down Menu	
Multi-select List Box	Drop-Down Menu	Allow Multiple Selections to Yes
Generic Button	Push Button	Button type to Normal
Submit Button	Push Button	Button type to Submit
Reset Button	Push Button	Button type to Reset

The FrontPage Editor adds the control at the location of the insertion point. If the insertion point is inside an HTML form, the control will be added to the form. If the insertion point is outside of a form, a new form will be created when you insert the control.

To see a demonstration of how to add standard controls to an HTML page, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

For information about the HTML syntax for standard controls, see the technical article, Syntax for Standard HTML Controls in the Articles and White Papers section of the Library.

Text box control

Check box control

Drop-down list box control

Radio buttons

Submit button

Reset button

Active Server Pages are processed by an extension on the Web server. This extension ships with Microsoft Visual InterDev and Microsoft Internet Information Server (IIS).

Before you can test Active Server Pages, the extension must be installed on a Web server. Because a Web server runs a process to read Active Server Pages, they must also be placed in a virtual folder that has Execute permissions.

An Active Server Page script runs when a browser requests an .asp file from a Web server. The Web server calls Active Server Pages, which runs scripts and commands, and then sends an HTML page to the browser.

The following illustration shows how Active Server Pages are processed between the Web browser and the Web server.

{ewc MVIMG, MVIMAGE,!W02G020.bmp}

Static HTML pages display the same information each time a user views the page in a browser, while Active Server Pages are dynamic and interactive. Whenever a user requests an .asp file, the script runs and returns an HTML response to the user.

To see a demonstration of how to view an Active Server Page with a browser, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Server-side script is contained in either <% %> delimiters, or in a <SCRIPT> tag with the RUNAT attribute set to Server.

The default language for Active Server Pages is Visual Basic Scripting Edition (VBScript), but you can change the default language to JScript by clicking **Options** on the **Tools** menu.

The <% %> Syntax

When a Web server processes Active Server Pages, it runs any script code between the <% and %> delimiters. You can add any valid VBScript code between these delimiters.

In general, an Active Server Page contains HTML code mixed with server-side script. The server-side script programmatically determines what information will be returned to the user. In an Active Server Page, Visual InterDev highlights server-side script in yellow to distinguish it from HTML code.

To see an illustration of how Visual InterDev uses yellow highlighting for server-side script on an Active Server Page, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

In the following example script, the Now and Hour procedures contained in the <% and %> delimiters determine the current time, and then greet the user with either "Good Morning" or "Good Day," depending on the time.

```
<% if Hour(Now) < 12 then %>
Good Morning.
<% else %>
Good Day.
<% end if %>
```

If it is 8:00 A.M., the HTML returned to the user will be:

```
Good Morning.
```

To display HTML to the user with output from the script, use the <%= %> syntax. For example, the following script displays the current time to the user:

```
The time here is now <%= Time %>.
```

If it is 8:34 A.M., the HTML returned to the user is:

```
The time here is now 8:34 AM.
```

The <SCRIPT> Syntax

You can also add server-side script to an HTML <SCRIPT> tag by setting the RUNAT attribute to Server. In a <SCRIPT> section, you can create server-side functions and **Sub** procedures that can be invoked from other script on the page.

In the following example code, the server-side script determines whether it is morning or afternoon:

```
<SCRIPT LANGUAGE=VBScript RUNAT=SERVER>
Function ComputeAMPM()
    If Hour(Now) < 12 Then
        ComputeAMPM = "morning"
    Else
        ComputeAMPM = "afternoon"
    End If
End Function
</SCRIPT>
```

In the following example, you can call the function from server-side script in <% and %> delimiters to display the following result:

```
Time for your <%= ComputeAMPM() %> classes.
```

You can display the result directly from the HTML <SCRIPT> section by using the **Response.Write** method, as follows:

```
<SCRIPT LANGUAGE=VBScript RUNAT=SERVER>
Response.Write "Time for your " & ComputeAMPM() & " classes."
</SCRIPT>
```

In the <SCRIPT> section, any code that is not contained in a procedure will run when the Web server processes the .asp file. Code in a procedure will not run until the procedure is explicitly invoked by server-side script.

In either scenario, if it is 6:00 A.M., the HTML returned to the user will be:

```
Time for your morning classes.
```

For more information about .asp files and writing server-side script, see Chapter 6: Using Active Server Pages.

Similar to ActiveX controls, initial property values define how a Java applet will appear when the HTML page is first downloaded by the browser.

To set initial properties for Java applets, you use <PARAM> tags.

The following example code shows the syntax of <PARAM> tags:

```
<PARAM NAME=appletParameter1 VALUE=value>
```

To specify the file that contains information for the Java Outline applet, you set the **TOCFile** property, as shown in the following example code:

```
<APPLET
    CODE=Outline.class
    WIDTH=150
    HEIGHT=200 >
    <PARAM NAME=TOCFile VALUE="contents.toc">
</APPLET>
```

If the .class files for a Java applet are located in a different folder than the HTML page, you must add the CODEBASE attribute to an <APPLET> tag. The value of the CODEBASE attribute can be an absolute or a relative URL.

The following example code shows the <APPLET> tag for the Outline Java applet in the relative folder Applets/Javaapps:

```
<APPLET
    CODE=Outline.class
    CODEBASE="applets/javaapps"
>
</APPLET>
```

Microsoft Internet Explorer and ActiveX controls can be run on almost any platform, such as the Intel x86, the Apple Macintosh, and any of several RISC computers. To make sure that your Web application can be run on various platforms, it is important to create, test, and package the files for each control, and distribute the correct files for a user's computer.

You can set the CODEBASE attribute to point to one of three file types.

® **Portable executable file**

This is a single executable file, such as an .ocx, .dll, or .exe file, that is downloaded, installed, and registered by the browser.

® **Cab (cabinet) file**

A .cab file contains one or more files, such as the .ocx, .dlls, and.inf file. The files included in the .cab file are downloaded together in a compressed form.

® **Inf file**

An .inf file contains script that specifies which files to download and set up in order for the .ocx file to run.

Note The CODEBASE attribute should not point to an .inf file because it cannot be signed. Instead, a .cab file should be created to contain only the .inf file.

Platform Independence

You can create platform-independent Setup scripts that retrieve files from different locations, depending on the platform. The following example code shows an .Inf file that supports the platforms Intel x86, MIPS, and Apple MacIntosh.

```
[Add.Code]
circ3.ocx=circ3.ocx
[circ3.ocx]
file_win32_x86=file://products/release/circ3/x86/circ3.cab
file_win32_mips=file://products/release/circ3/mips/circ3.cab
file_mac_ppc=file://products/release/circ3/macppc/circ3.cab
clsid={9DBAFCCF-592F-101B-85CE-00608CEC297B}
FileVersion=1,0,0,143
```

The file_%opersys%_%cpu = syntax enables the .Inf file to specify multiple locations where platform-dependent modules can be found and downloaded.

For more information about .Cab and .Inf files, jump to the CAB article on theMicrosoft ActiveX SDK Web site by clicking this icon:
{ewc mvimg, mvimage,!intjump.bmp}

On the Web site, download the CABinet Development Kit.

For more information on packaging MFC Controls, please see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q157959**
TITLE: How to Package MFC Controls for Use Over the Internet

Packaging Options

There are a number of different ways you can package an ActiveX control for different platforms. The following list provides several of these methods, and discusses the advantages and disadvantages of each.

® Create one .cab file that contains only an .inf file. Include locations for specific operating system/processor binaries in the .inf file.

This method provides a reduced download time because only the .inf file and the required binary files are downloaded. This is the preferred Internet solution for cross-platform support.

The disadvantage is that two digital certificates will be displayed to the user: one for the .cab file containing the .inf file, and another for the platform-specific .cab file.

® Create one .cab file that contains the .inf file and the Intel x86 binary file. Package the remaining binary files in separate .cab files, and include their locations in the .inf file.

This method provides the quickest download time for Intel x86 users. It is the preferred Internet solution for controls that are targeted for the x86 platform.

The disadvantage is that Apple Macintosh and RISC users must wait for the x86 binary files to be downloaded.

® Package all binary files into one .cab file.

This method provides the simplest package for small controls that do not rely on any other files. It is the preferred intranet solution when bandwidth is not a consideration.

The disadvantage is that download time is increased and that unnecessary files are downloaded.

® Use one compiled .ocx file for each operating system/processor.

This method is only applicable to small controls that have do not have any DLL dependencies. It provides the quickest download time, and is a possible solution for intranets where an administrator is confident of all users' hardware.

The disadvantage is that all users are required to work on the same platform. If users jump to the Web page using a different platform, they will not be able to use the controls unless they have already been installed.

When you use the FrontPage Editor to add standard controls to an HTML document, an HTML form will be inserted automatically. An HTML form is represented by the <FORM> tag.

In the following example code, a form contains standard HTML controls:

```
<FORM METHOD=POST>
    Email name: <INPUT NAME="txtEditBox" VALUE="My Name"><P>
    Check all that apply:
    <INPUT TYPE="CHECKBOX" NAME="chkBusinessUse"> Business use
    <INPUT TYPE="CHECKBOX" NAME="chkHomeUse"> Home use<P>
    <INPUT TYPE=SUBMIT VALUE="Submit">
    <INPUT TYPE=RESET VALUE="Reset">
</FORM>
```

Forms can contain any HTML elements except other forms. You can add more than one HTML form to a document, however, forms cannot be nested.

HTML forms package the names and values of each control, and then send them to the location specified by the ACTION attribute. The location can be a CGI application, an ISAPI application, or an Active Server Page.

In the following example code, the form will send information to the file Events.asp.

```
<FORM ACTION=events.asp METHOD=POST>
```

In FrontPage, the ACTION attribute is referred to as a form handler. To set a handler for a form, right-click anywhere in the form, and then click **Form Properties**.

To see an illustration of the **Form Properties** dialog box, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

FrontPage also includes a number of WebBot components that you can use as form handlers. A WebBot component is a dynamic object that generates HTML.

For information about WebBot components, go to the documentation for FrontPage on the Microsoft Web site (www.microsoft.com) by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

To specify your own form handler, such as an Active Server Page, use the **Form Properties** dialog box to set **Form Handler** to **Custom ISAPI, NSAPI, or CGI Script**, and then click **Settings**.

Sending Control Values to a Server

To send values of a control to the Web server, place a **Submit** button on the form. Only controls with the NAME attribute will be sent to the server.

Note Microsoft Internet Explorer does not require that all standard controls be placed on forms. However, if you want to send the information from controls to the server, you must use a form.

Only standard HTML controls are submitted with a form. To submit the value of an ActiveX control or Java applet with a form, set the VALUE attribute of a standard HTML control to an appropriate property of the ActiveX control or Java applet.

Typically, you use hidden controls to submit values of ActiveX controls or Java applets with a form. You create a hidden control in the form, and then add client-side script to the OnSubmit event procedure for the form.

In the OnSubmit event procedure of the form, you set the VALUE attribute of the hidden control to an appropriate property of the ActiveX control or Java applet. A hidden control is a standard HTML control, so the value of the control will be submitted with the other HTML controls on the form.

For more information about adding client-side script for controls, see Chapter 5: Adding Client-Side Script.

The easiest way to obtain information from a user and send it to a Web server is to use an HTML form.

An HTML form contains standard HTML controls, which are also referred to as intrinsic controls. These controls are supported by all Web browsers. Standard controls include text boxes, command buttons, radio buttons, and drop-down list boxes.

By using these controls, users can enter information, and then send the information to be processed by a Web server.

In this section, you will learn how to add standard HTML controls and forms to an HTML page by using the FrontPage Editor.

This section includes the following topics:

In this section, you will learn about Active Server Pages and how to add them to your Web project. You will also learn how to author server-side scripts.

For more information about adding server-side scripts to your Web site, see Chapter 6: Using Active Server Pages.

To see an animation that shows you how server-side script and components are used on both the client and the server, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

Active Server Pages have an .asp file extension and contain a combination of server-side script (either in the scripting language Visual Basic Scripting Edition or JavaScript) and HTML tags. The files are processed on a Web server.

By using server-side script, you can perform a number of tasks, such as gathering information from a user, retrieving data from a database, or building dynamic HTML pages that are returned to the user.

This section includes the following topics:

® Active Server Page Architecture

® Creating Active Server Pages

® Active Server Page Syntax

In stand-alone applications, there are specific starting and ending points at which you can write initialization and clean-up code. You can now use the same type of start/end paradigm in Web applications.

When you create a new project, Visual InterDev automatically creates a Global.asa file and adds it to the project under the project name. In the Global.asa file, you can create event procedures that run when your Web application starts or ends, or when a new session starts or ends.

Application and Session Information

A Web application can track information for both a Web application and a Web session.

- A Web application will start when a user first navigates to a page on a Web site. It will end after the last user has left the site. As long as one user is on the site, the application will still be active.
- A Web session will start when a user first navigates to a page on a Web site. A session ends automatically if the same user has not requested or refreshed a page on the Web site for a predetermined period of time. By default, the time is 20 minutes.

The **Application** object and **Session** object hold application and session information. The Global.asa file contains the OnStart and OnEnd event procedures for both objects.

You can add code to these event procedures to perform initialization or clean-up tasks, such as the following:

- Application_OnStart

  Runs when the application starts.
- Application_OnEnd

  Runs when the application ends.
- Session_OnStart

  Runs each time a user starts a new session, before the requested Web page is returned to the user.
- Session_OnEnd

  Runs each time a session is explicitly ended or times out. At this point, you cannot send output or take any other action.

While your Web application is running, you can retrieve information about the session or the application. Typically, you save user-specific information (such as user input on forms) in the **Session** object. The **Application** object contains application-wide information, such as information about data connections.

For more information about the Global.asa file and the **Application** and **Session** objects, see Chapter 6: Using Active Server Pages.

You can use Visual InterDev to create new Active Server Pages.

You can use either Visual Basic Scripting Edition (VBScript) or JScript as your server-side scripting language. To set the language preference for a page, click **Options** on the **Tools** menu, and then click the **HTML** tab. Under **Default Languages** for **Active Server Page**, click the scripting language you want to use.

To see a demonstration of how to create a new Active Server Page, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

u **To create new Active Server Pages**

1. On the **File** menu, click **New**.
2. On the **Files** tab, click **Active Server Page**.
3. In the **Add to project** box, add the .asp file to the current project.
4. In the **File name** box, type a name for the Active Server Page.
5. In the **Location** box, specify the location where you want the .asp file to be added, and then click OK.

   An Active Server Page template appears.

The first line of script sets the server-side language for the page. For example, if you set the language to VBScript, the following line of script will be added to the Active Server Page:

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

Note You cannot use the FrontPage Editor to edit an .asp file. To change an .asp file to an HTML file, add HTML script, and then change the file extension from .htm or .html to .asp. You can then use the Visual InterDev Source Editor to add the server-side script.

To add, modify, and delete records from a recordset, you can use methods of the **Recordset** object.

Note If you do not need to create a recordset, you can use a **Command** object and execute an SQL Insert, Update, or Delete statement to add or modify records. Using a SQL statement will be more efficient than creating a recordset and using recordset methods. For more information on using SQL statements, see Executing a Command in this chapter.

Adding Records

To add a record to a database, use the **AddNew** method of the **Recordset** object, set values for the record, and then use the **Update** method to save the record.

The following example code retrieves values from the **Request** object in an .asp file and adds a new record to the Majors table.

```
rsMajors.Addnew
    rsMajors("MajorID")= Request("MajorID")
    rsMajors("Description")= Request("Description")
rsMajors.Update
```

If you invoke **AddNew**, and then move from the current record, or you invoke another **AddNew** method before calling **Update**, ADO will automatically update the record. This behavior is different from DAO, which will discard the update.

To cancel an update, call the **CancelUpdate** method.

Updating Records

To change the values of the current record, set the appropriate fields and then use the **Update** method to save the changes.
{ewc mvimg, mvimage,!tip.bmp}

The following example code changes the description of the current record to Science, and saves the changes.

```
rsMajors("Description") = "Science"
rsMajors.Update
```

Deleting Records

To delete the current record, use the **Delete** method of the **Recordset** object. After you delete a record, invoke a **Move** method to move to the next record, otherwise the current record pointer points to an empty record.

The following example code deletes the current record in the rsStudents recordset.

```
rsStudents.Delete
rsStudents.MoveNext
```

To add ActiveX controls, you can use the FrontPage Editor as well as the Visual InterDev Source Editor.

u To add an ActiveX control with the FrontPage Editor

1. On the **Insert** menu, click **Other Components**, and then click **ActiveX Control**.

   To see an illustration of the **ActiveX Control Properties** dialog box, click this icon.
   {ewc mvimg, mvimage,!illust.bmp}
2. In the **ActiveX Control Properties** dialog box, set properties for the control.
   a. In the **Pick a Control** drop-down list box, select the control you want to add.
   b. In the **Name** text box, specify a name for the control.

      This adds the ID attribute to the <OBJECT> tag.
   c. In the **HTML** text box, specify alternate HTML that will be displayed by browsers that do not support ActiveX controls.
   d. In the **Code Source** text box, specify the Web location of the ActiveX control.

      This adds the CODEBASE attribute to the <OBJECT> tag. For information about the CODEBASE attribute, see Setting the CODEBASE Attribute.
3. To set options for the control, click **Properties**.

   This opens the ActiveX Control Editor, and a **Properties** dialog box containing options specific to the selected ActiveX control.
   a. To view the ActiveX control with the value of the options incorporated, click **Apply**.
   b. To close the ActiveX Control Editor, click OK.
4. To close the **ActiveX Control Properties** dialog box, click OK.

   FrontPage inserts the control into the HTML page.

u To edit an ActiveX control with the FrontPage Editor

1. Right-click the control you want to edit, and then select **ActiveX Control Properties**.

   This starts the ActiveX Control Editor for the selected object.
2. With the ActiveX Control Editor, edit the values of the control's properties.

   When you close the Object Editor, it updates the <OBJECT> tag for the control with the values.

To see a demonstration of how to use the FrontPage Editor to add and edit an ActiveX control on an HTML page, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

To complete the lab exercises in this chapter, you must have the required software. For detailed information about the labs and setup for the labs, see Labs in this course.

For background information on this lab, click each of these topics:

Objectives

Prerequisites

Lab Setup

To see a demonstration of the lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

To see a diagram of how the files you edit in this lab will fit into the State University Web site, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Estimated time to complete this lab: **60 minutes**

Note There are project and solution files associated with this lab. If you installed the labs during Setup, these files are in the folder <*Install Folder*>\Labs\Lab02 on your hard disk. If you did not install the labs during Setup, you can find them in the \Labs\Lab02 folder of the *Mastering Web Site Development* CD-ROM.

Exercises

The following exercises provide practice working with the concepts and techniques covered in Chapter 2: Developing a Web Project.

Exercise 1: Creating a New Project

In this exercise, you will create a new project for the State University Web site, and add existing HTML files to it.

Exercise 2: Creating a Static HTML Page

In this exercise, you will create a static HTML page that contains a table, hyperlinks, and images.

Exercise 3: Creating an Active Server Page

In this exercise, you will create an Active Server Page that outputs the date, time, and current class semester.

Exercise 4: Using a Form

In this exercise, you will create an HTML form that calls an Active Server Page and passes it a string.

After completing this lab, you will be able to:

® Create a new Visual InterDev project and add files to it.

® Create a new HTML page and add graphics and hyperlinks.

® Create frames on an HTML page.

® Create a new Active Server Page and call it from a hyperlink.

® Create an HTML form that calls an Active Server Page.

Before working on this lab, you should be familiar with the following concepts:

® Copying files between folders in Windows 95 or Windows NT.

® Browsing files with a Web browser.

® Adding HTML tags to a Web page.

To complete this lab, you need the following:

® Microsoft Visual InterDev version 1.0 client software.

® Visual InterDev server components installed on a Web server.

Data-aware Web pages enable users to perform database functions, such as searching for information, analyzing data, and updating a database.

There are two key technologies that you can use to build data-aware pages: ActiveX Data Objects(ADO) and the Advanced Data Connector (ADC).

To see a demonstration of how Web applications use ADO and ADC, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

ActiveX Data Objects

ActiveX Data Objects (ADO) is the next generation of data access models. It supersedes the models Data Access Objects (DAO) and Remote Data Objects (RDO). ADO is a collection of Automation objects that can retrieve, update, and create records in any OLE DB or ODBC database. ADO is installed with Microsoft Internet Information Server (IIS) version 3.0.

ADO runs in Active Server Pages on a Web server, and returns data from a database as HTML text. The returned page can be viewed by any browser on any platform. You can also use ADO in other host applications, such as Visual Basic and Visual C++.

Advanced Data Connector

The Advanced Data Connector (ADC) is a set of ActiveX controls that you can place on a Web page to enable a user to access a database. The ADC controls store records returned from a query as a recordset on the client workstation. The ADC displays data from the recordset to the user in data-bound controls, such as grids and list boxes.

With ADC, you can also write client-side script that moves through and updates the recordset locally without accessing the Web server again. A user can update several records, and then submit all changes back to the server in one update.

To use ADC, the browser must support ActiveX controls and client-side script. The Web server must also have server-side ADC components installed.

To install the Advanced Data Connector, go to the Microsoft ADC Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

Choosing a Technology

The Advanced Data Connector returns data to the client as a recordset object. The user can work with the data in the recordset without having to query the Web server again. ADC also serves as a data source for ActiveX controls.

ActiveX Data Objects run on the Web server and return static data to the client as HTML text. With ADO, you specify what type of recordset to create, which provides more flexibility than using ADC. ADO is also useful if you want to ensure that your application will work with any browser and any client.

When you create an application, you decide which technology is the most appropriate. You can use ADC, ADO, or a combination of both technologies in the same application.

If a recordset supports bookmarks (the cursor type is keyset or static), the **RecordCount** property of a recordset will return the number of records in the recordset.

If a recordset does not support bookmarks (the cursor type is forward only or dynamic), the **RecordCount** property will always return **-1**.

Some files for the State University Web site have already been created, but they have not been linked together in one location.

In this exercise, you will create a new Visual InterDev Web project, and then add existing HMTL files and graphics to it.

Create a new project

1. Start Visual InterDev and create a new project.
   a. On the **File** menu, click **New**.
   b. On the **Projects** tab, click **Web Project Wizard**.
   c. In the **Project name** box, enter StateU, and then click OK.
   d. In the Web Project Wizard, enter the name of your Web server, click **Next**, and then click **Finish**.
   {ewc mvimg, mvimage,!tip.bmp}
2. In Windows Explorer, locate the files that have been created on your computer and on the Web server.

   There are folders created on the Web server that have not been created on your computer, and vice versa.
   {ewc mvimg, mvimage,!tip.bmp}

Add files to the project

1. In Visual InterDev, add the image files in the folder \Labs\Lab02\images to the images folder in the StateU Web project.
2. In Visual InterDev, copy the files default.htm, crest.htm, home.htm, mascot.htm, and feedback.htm from the folder \Labs\Lab02 to the root directory of the StateU Web project.

Note The file mascot.htm uses a background image that you will add to the StateU Web project in Lab 3: Using Visual InterDev Data Tools. The appearance of this page will change after you have added items to the Web project.

---

3. In Visual InterDev, create a new folder named Controls in the root directory of the project.
4. Copy the file spin32.ocx from the folder \Labs\Lab02 to the Controls folder.

   For information about adding project files, see Adding Files to a Project.

View the home page and find the broken link

The home page for the State University Web site has three frames, as shown in the following illustration. To learn about the attributes of these frames, click each frame.

{ewc MVIMG, MVIMAGE,!W02G090.shg}

1. To view the State University home page, right-click the file default.htm in FileView, and then click **Preview in Browser**.
2. To find the broken link in default.htm, right-click the file default.htm in FileView, and then click **View Links**.
   {ewc mvimg, mvimage,!tip.bmp}

SRC = crest.htm
NAME = image

SRC = links.htm
NAME = links

SRC = home.htm
NAME = main

In this exercise, you will create an Active Server Page that will read student feedback about the State University Web site.

u Create an Active Server Page

1. In the StateU project, create a new Active Server Page named getfeedback.asp.
2. Open getfeedback.asp in the Visual InterDev Source Editor.
3. Add the following lines of HTML and server-side script between the <BODY> and </BODY> tags:

```
Your feedback was received at: <%=Now%>
<P>
<%For i=3 to 5 %>
<FONT SIZE=<%=i%>>
Today is <%=Date%> and it is <%=Time%>.<BR>
</FONT>
<%Next%>
```

4. Save your changes to getfeedback.asp, and then test the page by previewing it in the Visual InterDev InfoViewer.

   To see an illustration of how the page should look in the InfoViewer, click this icon.
   {ewc mvimg, mvimage,!illust.bmp}
5. To see the HTML generated by the server-side script, right-click the page while viewing it in the InfoViewer, and then click **View Source**.

   You will see the <FONT> tags and the current time and date generated by the server-side script.

The power of HTML is its ability to link text or images to another document. The browser highlights these elements to indicate that they are hyperlinks.

You can add hyperlinks to pages in your current project, to a page on an intranet, or to pages on the Web.

To create a hyperlink with the FrontPage Editor

1. Select the text that will identify the hyperlink.
2. On the **Insert** menu, click **Hyperlink**, or click the **Hyperlink** button on the toolbar.
3. In the **Create Hyperlink** dialog box, click the **World Wide Web** tab, and then specify the relative or absolute URL to which you want to link.

   For example, to create a hyperlink to the page Userdata.htm in your current Web project, select the text on the page, click the **Hyperlink** button, click the **World Wide Web** tab, and then type **userdata.htm** in the URL box.

   To see an illustration of the **World Wide Web** tab, click this icon.
   {ewc mvimg, mvimage,!illust.bmp}

   You can also create a hyperlink to a bookmark on the current page by clicking the **Open Pages** tab, and then selecting a bookmark from the **Bookmark** list.

HTML Syntax for Hyperlinks

The single hyperlink-related tag for HTML is <A>, which stands for anchor.

The following example code shows the HTML that the FrontPage Editor adds to a Web page when you create a hyperlink.

```
<A HREF="finance.htm">ABC Co. Financial Statement</A>
```

In this example, the syntax makes the text “ABC Co. Financial Statement” a hyperlink to the page Finance.htm, which is located in the same folder.

You can link to the following locations:

® Other folders on the same Web server, by specifying the relative path from the current document to the linked document, such as:

```
<A HREF="../file.htm">Link text</A>
```

This is a relative link.

® Other folders on the same Web server, by specifying the absolute path to the linked document, such as:

```
<A HREF="/folder/file.htm">Link text</A>
```

This is an absolute link.

® Other Web servers, by specifying the URL to the linked document, such as:

```
<A HREF="http://server/folder/file.htm">Link text</A>
```

Passing Information with a Hyperlink

You can also send information to a Web server directly from a URL by passing parameters from a Web page.

Each parameter consists of a name and a value. Parameters are appended to the URL with a question mark (**?**) after the name of the Web page.

If you pass more than one parameter, each name and value will be separated by an ampersand (**&**). If the value contains spaces, each space will be replaced by a plus sign (**+**).

The following example code shows how HTML adds question marks, ampersands, and plus signs when passing parameters.

```
<A HREF="/scripts/sample.dll?param1=value1&param2=value+2">
```

Another way to pass information to a Web server is through HTML forms. For information about how to create HTML forms and pass dynamic user data to server applications, see Creating HTML Forms.

With the exception of the selection and multiple-line text box controls, all HTML controls use the <INPUT> tag. The controls are distinguished by the TYPE attribute, which is different for each control type.

```
<INPUT TYPE=type of control NAME=name of control VALUE=value>
```

Note The NAME and VALUE parameters are common to all of the controls. Using the NAME parameter, you can assign a name to the control that uniquely identifies it within an HTML page. The VALUE parameter is the value assigned to the control when the user interacts with it. It is important to set these parameters if you plan to use the controls within a form or from VBScript. You will learn more about the role of the NAME and VALUE parameters, in Chapter 5.

Visual InterDev does not have a built in interface for adding standard HTML control. You can, however, edit the HTML file manually to insert the tags.

Text Box

The text box control enables the Web user to supply information as a string in the HTML page. There are two types of one-line text boxes:

® Single line text box presents a single line of text to type in.

```
<INPUT TYPE=TEXT NAME="txtName" VALUE="initial value" MAXLENGTH= SIZE= >
```

® Password text box presents a single line of text to type in, and masks what the user types with asterisks (*).

```
<INPUT TYPE=PASSWORD NAME="txtPassword" VALUE="">
```

® Multiple line text box allows the user to enter a block of text into the Web page.

```
<TEXTAREA ROWS=4 COLS=25>
put default text here..
</TEXTAREA>
```

Check Box

Use a check box control to present the user with a simple True or False question. The check box control is represented by the <INPUT> tag, with its TYPE attribute set to CHECKBOX. To add descriptive text to the check box, type the text on the HTML page after the INPUT tag.

This example displays a check box control that asks the user a question.

```
<INPUT TYPE=CHECKBOX NAME=chkFollowUp>Would you like a follow-up call?
```

Radio Button Group

A radio button (also called an option button) group presents the user with a number of mutually exclusive options. The user can select only one option from those included in the group. This control is added with the <INPUT> tag, with the TYPE attribute set to RADIO.

This example displays a radio-button group that provides three options: Yes, No, and Maybe.

```
<INPUT TYPE=RADIO NAME=optAskQuestion VALUE="Yes">Yes
<INPUT TYPE=RADIO NAME=optAskQuestion VALUE="No">No
<INPUT TYPE=RADIO NAME=optAskQuestion VALUE="Maybe">Maybe
```

Notice that the NAME parameter for each radio button is the same, indicating that the options are all members of the same group.

List Box

The list box control (also called the Selection control) is a drop-down list box that allows the user to select one or

more items from a number of possible options. The selection control is represented by the <SELECT> and </SELECT> tags, and each element in the list is indicated by the <OPTION> tag.

This example displays a selection control that contains a list of colors.

```
<SELECT NAME=lstFavoriteColor>
<OPTION VALUE="Red">Red
<OPTION VALUE="Blue" SELECTED>Blue
<OPTION VALUE="Yellow">Yellow
</SELECT>
```

To make it a multi-selection list box, add the MULTIPLE attribute to the SELECT tag.

Hidden

The hidden control enables you to attach static or unchanging data invisibly to the information sent with a form by the Web user. This information is sent to the server, along with the values of the visible controls. For example, you can use the hidden control to return the name of the HTML page that initiated the server call.

For the hidden control, the TYPE attribute of the <INPUT> tag is set to HIDDEN.

```
<INPUT TYPE=HIDDEN NAME=hdnName
VALUE="Information you don't want the user to see">
```

Button

The button control allows you to add an interface The VALUE parameter is the text displayed on the face of the button.

There are three types of buttons in the standard set of controls.

® Generic

The generic button does nothing unless you have assigned scripting code to run when it is clicked.

You can create a generic button by manually adding an <INPUT> tag to the HTML source code. For a generic button, the TYPE attribute is set to BUTTON, as in this example.

```
<INPUT TYPE=BUTTON NAME=btnClickMe VALUE="Click Me">
```

® Reset

The Reset button clears the contents of the form controls, restoring them to their default values. The reset button functions only when placed on a form with other controls.

The reset button has an <INPUT> tag, with the TYPE attribute set to RESET. The only other parameter that is supported by the reset button is VALUE, which corresponds to the text displayed on the button.

```
<INPUT TYPE=RESET VALUE="Reset">
```

® Submit

The Submit button sends the information the user has entered into the form field controls to the server. The submit button functions only when placed on a form with other controls.

The submit button has an <INPUT> tag, with the TYPE attribute set to SUBMIT. The only other parameter that is supported by the submit button is VALUE, which corresponds to the text displayed on the button.

```
<INPUT TYPE=SUBMIT VALUE="Go">
```

Forms can use three attributes to determine their behavior: ACTION, METHOD, and TARGET.

The ACTION Attribute

The ACTION attribute is the URL of the server extension that is invoked when the form is submitted to the server.

The METHOD Attribute

The METHOD attribute indicates the type of form-handling protocol used to send the form data to the server. The two possible values for the METHOD attribute are Get and Post.

® Get

If processing a form will not have any lasting effect on the state of the Web server, set the value of the METHOD attribute to Get. For example, database searches do not have a lasting effect on the state of a Web server.

With the Get method, the values of the controls are concatenated to the URL, and then passed to the server through HTTP with a data transmission limit of 1024K.

® Post

If processing a form does leave a lasting effect on a Web server, set the value of the METHOD attribute to Post. For example, modifying a database will leave a lasting effect on the state of the server.

With the Post method, the values of the controls are placed in the body of the HTTP request, so there is no limit to the number of parameters or length of the values.

The TARGET Attribute

The results of an HTML form are typically returned to the client's browser as a new HTML page. If the HTML page contains frames in the form, you can use the TARGET attribute to specify which frame should contain the results. For example:

```
<FORM TARGET="Frame1" ...>
```

1. What is a Visual InterDev Web project?

{ewc mvimg, mvimage,!answer.bmp} A. An autonomous Web site.

{ewc mvimg, mvimage,!answer.bmp} B. A container of files that create a Web site.

{ewc mvimg, mvimage,!answer.bmp} C. A container for .gif and .jpg files used in a Web site.

{ewc mvimg, mvimage,!answer.bmp} D. A container for .asp files used in a Web site.

2. How are Active Server Pages different from HTML pages?

{ewc mvimg, mvimage,!answer.bmp} A. Active Server Pages contain only server-side script; HTML pages contain only client-side script.

{ewc mvimg.mvimage.!answer.bmp}

B. HTML pages can contain <SCRIPT> sections; Active Server Pages can only contain in-line script.

{ewc mvimg.mvimage.!answer.bmp}

C. Active Server Pages are processed by the server before being returned to the client; HTML pages are sent to the client without being processed.

{ewc mvimg.mvimage.!answer.bmp}

D. Active Server Pages can be edited with the FrontPage Editor; HTML pages can be edited with the Visual InterDev Source Editor.

3. What does this server-side script return to the user's browser?

```
<% i = 5 %>
<%=i%> + <%=i%> = <%=i+i%>
```

{ewc mvimg.mvimage.!answer.bmp}

A. 5 + 5 = 5 + 5

{ewc mvimg.mvimage.!answer.bmp}

B. i + i = i+i

{ewc mvimg.mvimage.!answer.bm

p}

{ewc mvimg, mvimage,!answer.bmp} C. 5 + 5 = 10

{ewc mvimg, mvimage,!answer.bmp} D. i + i = 10

4. Which HTML tag would you use to create a nested frame?

{ewc mvimg, mvimage,!answer.bmp} A. <FRAMESET>

{ewc mvimg, mvimage,!answer.bmp} B. <FRAME>

{ewc mvimg, mvimage,!answer.bmp} C. <IFRAME>

{ewc mvimg.mvimage.!answer.bmp}

D. None of the above

5. What is the purpose of using forms?

{ewc mvimg.mvimage.!answer.bmp}

A. To contain license information about controls on a Web page.

{ewc mvimg.mvimage.!answer.bmp}

B. To group standard HTML controls visually on a Web page.

{ewc mvimg.mvimage.!answer.bmp}

C. To place controls in a file so they can be included with the same layout in other Web pages.

{ewc mvimg.mvimage.!answer.bmp}

D. To send information to a Web server.

6. Which of the following is not a standard HTML control?

{ewc mvimg, mvimage,!answer.bmp}

A. Multi-select list box

{ewc mvimg, mvimage,!answer.bmp}

B. Frame

{ewc mvimg, mvimage,!answer.bmp}

C. Check box

{ewc mvimg, mvimage,!answer.bmp}

D. Command button

7. Which type of file can you edit with the FrontPage Editor?

{ewc mvimg, mvimage,!answer.bmp}

A. .gif file

{ewc mvimg.mvimage.!answer.bmp}

B. .asp file

{ewc mvimg.mvimage.!answer.bmp}

C. .ocx file

{ewc mvimg.mvimage.!answer.bmp}

D. .htm file

Forms can be used to gather information from users and to create custom responses.

In this exercise, you will edit a form that has been created to gather feedback about the State University Web site, and then send the feedback to the Active Server Page, getfeedback.asp.

Note The page feedback.htm uses a background image and style sheet that you will add to the StateU Web project in Lab 3: Using Visual InterDev DataTools. The appearance of this page will change after items have been added to the Web project.

---

u **Call getfeedback.asp from a form**

1. With the FrontPage Editor In Visual InterDev, open feedback.htm.
2. Add a **Submit** button to the page by clicking **Form Field** on the **Insert** menu, and then clicking **Push Button**.

   The default button is a **Submit** button, so you do not need to edit the properties of the button.

   For information about adding controls to a Web page, see Adding Standard HTML Controls.

   The following illustration shows how the form should look.

   {ewc MVIMG, MVIMAGE,!W02G190.bmp}
3. Edit the properties of the form.

   a. Set **Form Handler** to **Custom ISAPI, NSAPI, or CGI Script**.

   b. Click **Settings**, and then set the ACTION attribute to getfeedback.asp and the METHOD attribute to Post.

   For information about adding forms to a Web page, see Adding HTML Forms.
4. Save your changes to feedback.htm, and then test the form by opening the page in the browser and clicking the **Submit** button.

1. Which ActiveX Data Objects (ADO) objects can you create with the CreateObject method?

{ewc mvimg, mvimage,!answer.bmp}

A. Only the **Connection** object.

{ewc mvimg, mvimage,!answer.bmp}

B. Only the **Connection** and **Command** objects.

{ewc mvimg, mvimage,!answer.bmp}

C. Only the **Command** and **Recordset** objects.

{ewc mvimg, mvimage,!answer.bmp}

D. The **Connection**, **Command**, and **Recordset** objects.

2. Assume that conn is a valid Connection object, and rsStudents is a valid Recordset object. Which statement will retrieve all of the records from the Students table?

{ewc mvimg, mvimage,!answer.bm

A. Set rsStudents = conn.OpenRecordset "select * from students".

p}

{ewc mvimg, mvimage,!answer.bmp}

B. rsStudents.OpenRecordset "select * from students", conn

{ewc mvimg, mvimage,!answer.bmp}

C. rsStudents.Open "select * from students", conn

{ewc mvimg, mvimage,!answer.bmp}

D. Set rsStudents = CreateObject ("ADODB.Recordset", "select * from students", conn)

3. How is an ADO database error handled in an Active Server Page that does not have an error handler?

{ewc mvimg, mvimage,!answer.bmp}

A. The error is ignored and processing continues. No message is returned to the user.

{ewc mvimg, mvimage,!answer.bmp}

B. A Web page containing nothing but the error message from the database is returned to the user.

{ewc mvimg.mvimage,!answer.bmp}

C. The Web browser displays a dialog box with the message "Error processing Web page."

{ewc mvimg.mvimage,!answer.bmp}

D. The error message from the database is included at the bottom of the Web page returned to the user.

4. When you use the Advanced Data Connector (ADC) to retrieve records, where are the records cached?

{ewc mvimg.mvimage,!answer.bmp}

A. On the Web client.

{ewc mvimg.mvimage,!answer.bmp}

B. In a business object on the Web server or Web client.

{ewc mvimg.mvimage,!answer.bmp}

C. On the Web server.

{ew

D. On the database server.

{ewc mvimg, mvimage,!answer.bmp}

5. When you use the AdvancedDataControl object, how do you specify where the data will be displayed?

{ewc mvimg, mvimage,!answer.bmp}

A. Set the BINDINGS attribute of the **AdvancedDataControl** object to the names of the data-bound controls in which the data will be displayed.

{ewc mvimg, mvimage,!answer.bmp}

B. Set the DATASOURCE attribute of the data-bound controls in which the data will be displayed to the name of the **AdvancedDataControl** object.

{ewc mvimg, mvimage,!answer.bmp}

C. Set the BINDINGS attribute of the data-bound controls in which the data will be displayed to the name of the **AdvancedDataControl** object.

{ewc mvimg, mvimage,!answer.bmp}

D. Write client-side script for a Web page.

When a user clicks a hyperlink in a frame, the link loads (or displays) in the target frame.

When you create a hyperlink, you can change the default target frame where a link should be loaded by using the TARGET attribute of the <A> tag.

The following table lists and describes the values of the TARGET attribute.

Attribute	Description	Example
"window"	Sets the link to load into a specific window. In the <FRAMES> tag, this is the NAME attribute of the frame you want to replace.	<A TARGET="frame1" HREF="sample.htm">
"_blank"	Sets the link to load into a new blank window. The window is not named.	<A TARGET="_blank" HREF="sample.htm">
"_parent"	Sets the link to load into the parent window of the window in which the link is located.	<A TARGET="_parent" HREF="sample.htm">
"_self"	Sets the link to load into the same window in which the link was clicked. This is the default.	<A TARGET="_self" HREF="sample.htm">
"_top"	Sets the link to load into the entire window.	<A TARGET="_top" HREF="sample.htm">

You can also specify that the target frame for loading all hyperlinks on a page should be in the same location. To set this location, you use the <BASE> tag with the TARGET attribute.

In the following example code, the <BASE> tag specifies that all links should be loaded in the body of a window.

```
<BASE TARGET="_top">
```

Note With the FrontPage Editor, you set the target frame in which a link should be loaded by using the **Create Hyperlink** dialog box. To set the base target frame for all hyperlinks on a page, you use the **Page Properties** dialog box.

When you develop a Web project, you follow a few basic steps. These steps are:

1. Retrieving files from the Web server to your local development computer.
2. Opening the files for editing.
3. Saving changes to the local files.
4. Submitting the files back to the Web server.
5. Previewing the changed files in a Web browser.

You can perform all of these steps in Microsoft Visual InterDev.

Retrieving Files from the Server

The Web pages on a Web site are stored on a Web server. To edit Web pages, you retrieve copies of the files from the Web server, work with the files in a Visual InterDev project on your local development computer, and then send the files back to the server.

Each member of the development team creates their own Visual InterDev project and edits separate copies of the Web pages.

If you want to open a file for editing, you can choose one of the following options:

® Double-click the file, or right-click the file and then click **Open**.

When you click the **Open** command, Visual InterDev retrieves a working copy of the file from the Web server, and opens the file with the Visual InterDev Source Editor. If the file on the server has been changed since it was last saved on your local computer, the **Confirm Update** dialog box provides the option of opening the most recent version of the file.

® Right-click the file, and then click **Open With**.

When you click the **Open With** command, the **Open With** dialog box enables you to select the editor or viewer you want to use to open the file with. After you choose the editor, a working copy of the file will be retrieved from the Web server and opened with that editor. If the file on the server has been changed since it was last saved on your local computer, the **Confirm Update** dialog box provides the option of opening the most recent version of the file.

® Click **Open** on the **File** menu.

This command opens the local version of a file. If a more recent version of the file is on the Web server, this command will not retrieve that working copy.

If you want to retrieve only the most recent version of a file from the Web server, but not open it for editing, right-click the file in FileView, and then click **Get Working Copy**.

Saving Changes to Files

To save changes to local files and to the Web server, click the **Save** button on the Visual InterDev toolbar, or click **Save** on the **File** menu.

To change this default behavior of saving locally and to the server, click **Options** on the **Tools** menu, and then click the **Web Projects** tab. Under **Project Options**, clear the check box **Update server when files are saved**.

To remove the local copy of the file from your local computer, right-click the file, and then click **Release Working Copy**.

Previewing File Changes in the Browser

To see how a file will be displayed to the user, right-click the file in your Visual InterDev project workspace, and then click **Preview in Browser**. The file will open in the InfoViewer, which is the default browser for Visual InterDev.

To view the file with a different browser, right-click the file, and then click **Browse With**. You can then can select another browser from the **Browse With** dialog box.

To change the default browser

1. Right-click a file in the project, and then click **Open With**.
2. In the **Open With** dialog box, select the browser you want to set as the default.
3. Click the **Set as Default** button.

While FrontPage has a good WYSIWYG editor for creating tables, it's useful to be able to read and edit the HTML tags directly to slightly alter the way the table looks in an Internet browser.

There are many effects you can create with tables using HTML 3.0; they are not all covered here. To see all of the table tags supported by HTML 3.0, refer to the HTML Reference page on the Microsoft Web site:
{ewc mvimg, mvimage,!intjump.bmp}

For a tutorial on using tables, see the Internet Explorer 3.0 Enhanced Tables: Step By Step page on the Microsoft Web site:
{ewc mvimg, mvimage,!intjump.bmp}

Basic Table Tagging

A table is created by entering the <TABLE>and </TABLE> tags in an HTML document. Any attributes that apply to the table as a whole are entered in the <TABLE> tag:

```
<TABLE ALIGN=LEFT BORDER=1 WIDTH=20%>

</TABLE>
```

Each row in the table is created by using the <TR></TR> tags. With each row, columns are created with the <TD></TD> tags. Any HTML element can be placed within the <TD></TD> tags (for example, text, images or image maps, hyperlinks, or forms). This example creates a three-row by three-column table on the page.

```
<TABLE ALIGN=LEFT BORDER=1 WIDTH=20%>
<TR><TD>R1 C1</TD><TD>R1 C2</TD><TD>R1 C3</TD></TR>
<TR><TD>R2 C1</TD><TD>R2 C2</TD><TD>R2 C3</TD></TR>
<TR><TD>R3 C1</TD><TD>R3 C2</TD><TD>R3 C3</TD></TR>
</TABLE>
```

The resulting table looks like this:

{ewc MVIMG, MVIMAGE,!W02G400.bmp}

Table Width and Height

You can set the width and height of the cells in your table by placing the WIDTH and HEIGHT attributes within the <TD> tag. If you create a table using Internet Assistant for Word, and move the column borders with the mouse, the WIDTH attribute is entered for you.

If you do not include the WIDTH or HEIGHT attribute in the tag, the table will collapse and expand with the size of the browser window. The width (or height) of each column (or row) will be based on the largest element in that row or column.

An alternative to this is to set the WIDTH or HEIGHT attributes to a percentage value, rather than a fixed size. For example, if you set the WIDTH of the <TABLE> tag to 50%, the table will always take up 50% of the available width, as determined by the size of the Web browser window. The table will collapse and expand with the browser. The preceding table example will always expand or collapse to include 20% of the available browser width.

Spanning Columns or Rows

You can use the COLSPAN and ROWSPAN attributes of the TD and TH tags to extend the contents of a cell into adjoining cells. This is useful if you need to stretch a column heading across more than one column.

This example spans and centers the contents of the first column in the first row across the top of the table. The contents of the first column in the second row are spanned down to include the third row in the table:

```
<TABLE ALIGN=LEFT BORDER=1 WIDTH=20%>
<TR><TD COLSPAN=3 ALIGN=CENTER>R1 C1</TD></TR>
<TR><TD ROWSPAN=2>R2 C1</TD><TD>R2 C2</TD><TD>R2 C3</TD></TR>
<TR><TD>R3 C2</TD><TD>R3 C3</TD></TR>
```

```
</TABLE>
```

The resulting table appears as follows:

{ewc MVIMG, MVIMAGE,!W02G410.bmp}

{ewc mvimg, mvimage,!tip.bmp}

Table Headers

You can specify a header (or bold text) for your table. Any cell that uses the <TH></TH> tag in place of the <TD></TD> tags will display bold text in your table.

```
<TABLE ALIGN=LEFT BORDER=1 WIDTH=20%>
<TR><TH COLSPAN=3 ALIGN=CENTER>R1 C1</TH></TR>
<TR><TD ROWSPAN=2>R2 C1</TD><TD>R2 C2</TD><TD>R2 C3</TD></TR>
<TR><TD>R3 C2</TD><TD>R3 C3</TD></TR>
</TABLE>
```

The resulting table appears as follows:

{ewc MVIMG, MVIMAGE,!W02G420.bmp}

Table Borders

You can use the BORDER attribute to create a border for your table. The number specified sets the width of the border.

```
<TABLE BORDER=4>
```

The FRAME and RULES attributes of the TABLE tag let you control how the table border is drawn. The FRAME attribute specifies which sides of a frame (outer borders) are displayed. The RULES attribute specifies which dividing lines (inner borders) are displayed.

This example displays a border around the outside of the table and horizontal rules separating the rows:

```
<TABLE BORDER=2 FRAME=box RULES=rows>
```

The resulting table appears as follows:

{ewc MVIMG, MVIMAGE,!W02G450.bmp}

Table Caption

You can add a caption to your table using the CAPTION attribute. Use the ALIGN attribute to set the horizontal alignment of the caption. Use the VALIGN attribute to set the vertical position of the caption (TOP or BOTTOM).

This example adds a caption that is centered at the bottom of the table.

```
<TABLE ALIGN=LEFT BORDER=1 WIDTH=20%>
<CAPTION ALIGN=CENTER VALIGN=BOTTOM>My Table</CAPTION>
<TR><TH COLSPAN=3>R1 C1</TH></TR>
<TR><TD ROWSPAN=2>R2 C1</TD><TD>R2 C2</TD><TD>R2 C3</TD></TR>
<TR><TD>R3 C2</TD><TD>R3 C3</TD></TR>
</TABLE>
```

Grouping Columns or Rows

The <COLGROUP> tag provides a way to specify attributes for an entire column in a table. The order of the <COLGROUP> tags determines which column a tag refers to.

These tags center the first column, left-align the second column, and center the third and fourth columns in a table:

```
<COLGROUP ALIGN=CENTER>
<COLGROUP ALIGN=LEFT>
<COLGROUP ALIGN=CENTER SPAN=2>
```

Nesting Tables

Tables can be nested together to create more complex tables. To nest one table within another, insert another <TABLE> and </TABLE> tag in place of text or other cell content. For example:

```
<TABLE ALIGN=LEFT BORDER=1 WIDTH=20%>
<TR><TH COLSPAN=3 ALIGN=CENTER>R1 C1</TH></TR>
<TR><TD ROWSPAN=2>
    <TABLE BORDER 1>
    <TR><TD>A</TD><TD>B</TD></TR>
    <TR><TD>C</TD><TD>D</TD></TR>
    </TABLE>
</TD><TD>R2 C2</TD><TD>R2 C3</TD></TR>
<TR><TD>R3 C2</TD><TD>R3 C3</TD></TR>
</TABLE>
```

The resulting table appears as follows:

{ewc MVIMG, MVIMAGE,!W02G430.bmp}

Sever-side includes give you the ability to include the same HTML tags in many files without having to copy and paste into each file. Besides saving you design time, this also gives you the advantage of making changes in one file and having the changes reflected in any other file. (If you are familiar with the #include construct in C and C++, server-side includes work in the same way.)

u To use a server-side include in a Web page

1. Create the include file. This file will contain the HTML tags that you want duplicated across other files.
2. Save the file on the server in the same directory as the files that will use the include file. (This is not necessary, but it is recommended.) The file should have an .htm extension.
3. In each HTML document that will use the include file, insert a comment that contains the string `#include file="/includefile.htm"`. For example:

   ```
   <!--#include file="/includefile.htm" -->
   ```

   Change the extension of the HTML document to .stm or .asp.

Note The required extension depends on how the Internet server was configured during Setup. By default, Internet Information Server is set up to associate files using server-side includes with the extension .stm. Furthermore, Internet Information Server with Active Server extensions will process server-side include statements in an .asp file.

When the server receives a request for a file with an .stm extension, it passes the file back to the client while the server looks for the Include statement. The server replaces the comment tag containing the #include string with the contents of the file specified in the FILE parameter.

Frames divide a Web page into multiple scrollable areas, with each area defined by its own HTML file. You can use frames to create visual effects on a Web page.

Here are some examples of how you can use frames on a Web page:

® In a static frame, you can place elements that you always want the user to see, such as a title banner and links. These elements will remain visible as the user scrolls through information on the page.

® In side-by-side frames, each frame is defined by a separate HTML file, so you can display a query on one side and the result on the other. This enables a user to move back and forth between the frames.

For more information about frames, go to the HTML Features page on the Microsoft Web site (www.microsoft.com) by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

This section includes the following topics:

® Adding Frames

® Nesting Frames

® Creating Hyperlinks in Frames

In an HTML file, tables can contain any valid HTML text, images, forms, or controls. With the FrontPage Editor, you can create a table with the **Insert Table** button, or insert table tags directly into the HTML file.

For information about the HTML tags used to create tables, see the technical article, HTML Table Tags in the Articles and White Papers section of the Library.

u **To create a table with the FrontPage Editor**

1. Click the location where you want the table to appear on the page.
2. On the **Table** menu, click **Insert Table**.
3. In the **Insert Table** dialog box:
   a. Select the number of rows and columns.
   b. Select an alignment.
   c. Select a border size.
   d. If you want the table to use a specific percentage of the page, or if you want it to be a fixed pixel size, select the **Specify Width** check box.

Note If there is not a specific reason for your table to have a fixed pixel size, clear the **Specify Width** check box. This enables you to select a percentage for the table, or enables the browser to select an appropriate width.

If you do not specify a width for the table, the FrontPage Editor will create minimum-size cells. As you type information into the table, the cells will expand automatically.

After you have inserted a table, you can use the **Table** menu to customize it. You can also change the properties for the table by right-clicking anywhere in the table, and then clicking **Table Properties**.

To see an illustration of the properties you can set in the **Table Properties** dialog box, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

To change the properties for individual table cells, right-click the cell, and then click **Cell Properties**.

To see an illustration of the properties you can set in the **Cell Properties** dialog box, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

The following table lists and describes some of the cell properties you can set.

Property	Description
Header Cell	Sets the font in the cell to bold.
Background Color	Sets the background color of the cell.
Rows Spanned	Sets the cell to span down more than one row.
Columns Spanned	Sets the cell to span across more than one column.

With the FRAMESET tags, an entire Web page is composed of information in frames. However, you can create a single frame that is displayed in an HTML page with the IFRAME tag. It is called a floating frame and can be located anywhere on a Web page. Place the <IFRAME> and </IFRAME> tags in the BODY of the page where you want the frame to appear, and set the WIDTH and HEIGHT attributes to specify the size of the frame.

For example, this HTML places a floating frame in the second row of a table:

```
<TABLE BORDER=2>
<TR><TD>Column 1</TD><TD>Column 2</TD><TD>Column 3</TD></TR>
<TR><TD COLSPAN=3>Row 2 with Floating Frame<P>
    <IFRAME SRC=source.htm WIDTH=200 HEIGHT=70></IFRAME>
    <P>Followed by more information...
</TD></TR>
</TABLE>
```

The floating frame in this example appear as follows:

{ewc MVIMG, MVIMAGE,!W02G055.bmp}

For more information on creating floating frames, see the Microsoft Floating Frames Step by Step Web site:
{ewc mvimg, mvimage,!intjump.bmp}

For an example of floating frames in action, see the Microsoft Floating Frames Example page:
{ewc mvimg, mvimage,!intjump.bmp}

To create a grid of frames on a Web page, you can nest <FRAMESET> tags.

The following example code creates a page with five frames, two in the first column, and three in the second column:

```
<FRAMESET COLS="40%,*">
    <FRAMESET ROWS="35%,*">
        <FRAME SRC="Cell_1.htm">
        <FRAME SRC="Cell_2.htm">
    <FRAME SRC="Cell_3.htm">
    </FRAMESET>
</FRAMESET>
```

The following illustration shows how these nested frames would appear in a browser.

{ewc MVIMG, MVIMAGE,!W02G045.bmp}

Each HTML page that you create with Visual InterDev has properties that you can set with the FrontPage Editor. These properties affect the appearance of the HTML page, provide information to browsers about how to handle the hyperlinks, and supply information about the page.

To open an HTML page with the FrontPage Editor, right-click the file name, and then click **Open With**. The **Open With** dialog box appears, where you can select the FrontPage Editor. To view the HTML tags that have been set on the page, click **HTML** on the **View** menu.

To set the properties of a page, click **Page Properties** on the **File** menu. Set the properties you want by clicking the **Background**, **Margins**, and **Custom** tabs.

The **General** tab provides information about the HTML page. To see an illustration of the **General** tab in the **Page Properties** dialog box, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

To set the background and colors for the page, click the **Background** tab. For example, to set the background image of an HTML page, select the **Background Image** check box. In the text box that appears when you select this option, type the absolute or relative URL for the image file.

The following table lists and describes some of the common properties you can set.

Property	Description
Title	A descriptive word or phrase that appears in a browser's title bar when the page is displayed.
Background Image or Watermark	An image that appears on a page behind the contents of the HTML page.
Background color	The color that appears behind the contents of the page.
Foreground color	The color of the text on the page.
Hyperlink color	The color of all hyperlinks on a page before they have been visited.
Background sound	A sound file that you associate with the page.

To complete the lab exercises in this chapter, you must have the required software. For detailed information about the labs and setup for the labs, see Labs in this course.

For background information on this lab, click each of these topics:

Objectives

Prerequisites

Lab Setup

To see a demonstration of the lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

To see a diagram of how the files you edit in this lab will fit into the State University Web site, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Estimated time to complete this lab: **60 minutes**

Note There are project and solution files associated with this lab. If you installed the labs during Setup, these files are in the folder <*Install Folder*>\Labs\Lab07.1 on your hard disk. If you did not install the labs during Setup, you can find them in the \Labs\Lab07.1 folder of the *Mastering Web Site Development* CD-ROM.

Exercises

The following exercises provide practice working with the concepts and techniques covered in Chapter 7: Creating Database-Aware Web Pages.

Exercise 1: Retrieving Records

In this exercise, you will create an Active Server Page that uses ADO to retrieve the classes and grades for the student.

Exercise 2: Adding Records

In this exercise, you will create an Active Server Page that obtains feedback information from a student and uses ADO to add the information to the Feedback table in the database.

Exercise 3 (Optional): Handling Database Errors

In this exercise, you will modify your .asp file to handle errors.

In this lab, you will use ADO code in Active Server Pages to retrieve and update a database. You will also modify existing .asp files. Optionally, you will add code to handle possible database errors.

After completing this lab, you will be able to use ADO to:

® Create a connection to a database.

® Create a **Recordset** object and retrieve records.

® Add a new record to a table.

Before beginning this lab, you should be able to:

® Create Web pages, .asp files, and projects in Microsoft Visual InterDev.

® Write server-side script in an Active Server Page.

® Call an ActiveX server component from an .asp file.

To complete this lab, you need the following:

® The Visual InterDev State University Project

® The State University Microsoft SQL or Microsoft Access Database

To complete the lab exercises in this chapter, you must have the required software. For detailed information about the labs and setup for the labs, see Labs in this course.

For background information on this lab, click each of these topics:

Objectives

Prerequisites

Lab Setup

To see a demonstration of the lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

To see a diagram of how the files you edit in this lab will fit into the State University Web site, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Estimated time to complete this lab: **60 minutes**

Note There are project and solution files associated with this lab. If you installed the labs during Setup, these files are in the folder <*Install Folder*>\Labs\Lab07.2 on your hard disk. If you did not install the labs during Setup, you can find them in the \Labs\Lab07.2 folder of this *Mastering Web Site Development* CD-ROM.

Exercises

The following exercises provide practice working with the concepts and techniques covered in Chapter 7: Creating Database-Aware Web Pages.

Exercise 1: Filling in a Data-Bound List

In this exercise, you will create a Web page that contains the **AdvancedDataControl** object and the data-bound list control **DBList**. The **AdvancedDataControl** object retrieves all State University classes from the database, and fills the **DBList** control with the titles of each class.

Exercise 2: Scripting the Advanced Data Factory

In this exercise, you will modify the file classview.htm (which you created in the previous exercise). You will use the objects **AdvancedDataSpace** and **AdvancedDataFactory** to fill the HTML text boxes on the HTML page with data.

After completing this lab, you will be able to:

- Use the **AdvancedDataControl** object to retrieve a recordset and display the data in a data-bound control.
- Create an instance of the **AdvancedDataFactory** object, and use methods of the object to retrieve a recordset.

Before starting this lab, you should be able to:

® Create Web pages, .asp files, and projects in Microsoft Visual InterDev.

® Write server-side script in an Active Server Page.

® Call an ActiveX server component from an .asp file.

To complete this lab, you need the following:

® Visual InterDev StateU Web project

® State University Microsoft SQL database or Microsoft Access database

® The Advanced Data Connector

® Visual Basic 5.0

State University students will be able to retrieve their transcripts online. Once they have entered their student ID through the profile.asp file, they can see a list of classes they have completed.

In this exercise, you will create the Active Server Page that uses the student ID from the session object to retrieve the classes and grades for that student. Using ADO, you will establish a connection to the State University database, retrieve the transcript for the student into a recordset, and then print the records into an HTML table.

To see an illustration of how the completed Active Server Page will look when returned to the student, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Add files to project

1. Using Microsoft Visual InterDev, open the StateU project.
2. In the \MWD\Labs\Lab07.1 folder, add the file getfeedback.asp to the root of the StateU project. This file replaces the one you created in Lab 2.

   For information about how to add files to a project, see Adding Files to a Project Chapter 2: Developing a Web Project.

Create the data connection

1. In the file transcript.asp, just before the <HTML> tag, add server script that retrieves the student ID from the session object and places it in a variable named frmStudentID.
2. On the line after the </TABLE> tag, type **<%** to begin entering server script.
3. Create a **Connection** object named conn, by using the **CreateObject** function.
4. Using the session values stored in the global.asa file for the State University connection, set the **ConnectionTimeout** and **CommandTimeout** properties for the object.
5. Invoke the **Open** method to establish the connection. Use the session values stored in the global.asa to set the connection string, user name, and password.

   For more information about creating a connection, see Establishing a Database Connection in this chapter.

   To see an example of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

Retrieve the transcript records

1. Use the **CreateObject** method to create a **Recordset** object variable named rsTranscript.
2. Set the **ActiveConnection** property of rsTranscript to the conn **Connection** object.
3. To retrieve the transcript records for the student ID, invoke the **Open** method of rsTranscript.

   The query should retrieve the classid, title, and grade fields from the Enrollment table and Classes table, where StudentID equals frmStudentID.

   To see an example of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

Write the transcript records to HTML

1. In the Active Server Page, after the server script from the previous procedure, use <TABLE> tags to create an HTML table that has three columns. In the first row, enter the column headers ClassID, Title, and Grade.

   To see an example of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}
2. After the first row of the table, enter server script that defines a **Do Until...Loop** statement to loop through the recordset. Use the **EOF** property of the recordset to determine when to exit the loop.
3. In the loop, write script that prints the ClassID and Title fields from the rsTranscript recordset. Use the beginning and ending <TD> tags around the fields to print the fields in the table columns.
4. To convert the Grade field from a number to a letter before printing, write a **Select Case** statement that converts 4, 3, 2, 1, 0 to A, B, C, D, and F, respectively, and then print the letter grade in the third column. Include a Case Else clause to check for a no grade.

5. In the last step of the loop, print HTML tags to start a new table row, and add server script to move to the next record in the rsTranscript recordset.
6. Add a link to links.htm that links the text Get Transcript to transcript.asp.

   To see an example of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

u **Test your solution**

® Save the .asp file and test it by previewing the transcript.asp file.
Note: The transcript.asp file uses a student ID provided on the login form. If you preview the transcript.asp before logging in with a valid student ID, you will receive an error message.

The State University Web site includes a Web page that students can use to submit comments about how well the StateU Web site is filling their needs.

In this exercise, you will create an Active Server Page that uses ADO to obtain feedback submitted by students, and add it to the Feedback table in the StateU database. You will use the design-time **Include** control to add the file adovbs.inc, which will enable you to use the ADO constants in the server script.

The feedback page (feedback.htm) is provided for you in this exercise. To see an illustration of how this feedback page looks, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

When a student submits the file feedback.htm, the file getfeedback.asp will run.

A starting point for the getfeedback.asp file has been provided for you. You will modify this file to retrieve the information from the form, update the database, and provide a confirmation message to the student.

Include the adovbs.inc file

1. In Visual InterDev, open the StateU project.
2. Add the adovbs.inc file to the root of the StateU project.

   Note The file adovbs.inc is installed in the \InetPub\ASPSamp by the Setup program of IIS 3.0 Active Server Pages. If you did not install the .asp sample files with IIS, you can copy the file adovbs.inc from the folder \MWD\Labs\Lab07.1.

   For information about how to add files to a project, see Adding Files to a Project in Chapter 2: Developing a Web Project.
3. Open the file getfeedback.asp to edit it in Visual InterDev.
4. On a blank line just above the <HTML> tag, insert the design-time **Include** control.

   For information about how to use design-time controls, see Using Design-Time Controls in Chapter 2: Developing a Web Project.
5. Set the source file of the **Include** control to adovbs.inc, and close the Properties Page and Object Editor.

   The information needed to include the control will be added to your file.
6. After the information for the **Include** control, add server script to obtain the form parameters from the **Request** object.
7. Store the four parameters that are passed, Response, Ease, Interactive, and Useful, in the variables **frmResponse**, **frmEase**, **frmInteractive**, and **frmUseful**, respectively.

   To see an example of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

Create the data connection

1. Create a **Connection** object named conn by using the **CreateObject** method.
2. Using the session values stored in the global.asa for the State University connection, set the **ConnectionTimeout** and **CommandTimeout** properties for the object.
3. Establish the connection, by invoking the **Open** method. Use the session values stored in the global.asa for the connection string, user name, and password.

   For information about how to create a connection, see Establishing a Database Connection in this chapter.

   To see an example of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

Add a new record

1. To add a new record, call the Execute method on the conn object to run an SQL query that inserts the form parameters into the database. The following sample code shows how your query should look.

```
strSQL = "INSERT INTO feedback " & _
   "(Response, Useful, Interactive, Ease)" & _
```

```
" VALUES (" & frmResponse & "," & frmUseful & _
"," & frmInteractive & "," & frmEase & ")"
```

To see an example of how your code should look, click this icon.
{ewc mvimg, mvimage,!code.bmp}

Test your solution

1. Save the file getfeedback.asp, and test it by previewing the feedback.htm file and submitting the form.
   A response page should be returned to you.
2. In Data View, open the Feedback table and verify that the new record has been added.

Error handling is an important part of writing robust Active Server Pages. In this exercise, you will modify the file getfeedback.asp to handle an error in adding a new record.

You will add a new file to the StateU project named error.asp. When an error is detected, getfeedback.asp will store the error information in session variables, and then redirect control to error.asp. The file error.asp will retrieve the error information from the session variables, display a message, and print the error title and error description in the HTML page.

This exercise demonstrates a simple error-handling approach for State University. However, in a real world application, you may want to use more powerful techniques, such as logging errors to a Web server log or by taking different actions based on the error.

Add files to the project

1. In Visual InterDev, open the StateU project.
2. Add the file error.asp from the folder \MWD\Labs\Lab07.1 to the root of the StateU project.

Check for errors when adding a new record

1. Open the file getfeedback.asp to edit it in Visual InterDev.
2. To change the default error-handling behavior, add the statement “On Error Resume Next” just before the statement that establishes a database connection.
3. Find the server script that adds the new record with the SQL insert statement.
4. After the **Execute** method is called to run the SQL insert, add an **If** statement to check if an error has occurred.
5. If an error has occurred:
   a. Set the ErrorTitle session variable to Feedback Form.
   b. Set the ErrorText session variable to the **Description** property of the **Err** object.
   c. Invoke the **Response.Clear** method to clear the HTML Response.
   d. Invoke the **Response.Redirect** method to redirect the .asp file to error.asp.

   For information about using the **Redirect** method of the **Response** object, see The Redirect Method in Chapter 6: Using Active Server Pages.

   To see an example of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

Test the error-checking code

1. Save the file getfeedback.asp.
2. Preview the file feedback.htm in the browser and submit it without clicking any radio buttons.

   This will force the parameters to be empty which will cause the SQL insert to fail. Verify that the error.asp page displays properly.

To create a line between two rows in a table, set the COLSPAN attribute to the number of columns in your table and set the BGCOLOR attribute to whatever color you want the line to be. To make the line display in the table, you can include the
 tag or a space ( ) on the line.

```
<TR><TD BGCOLOR="#808000" COLSPAN=4><BR></TD></TR>
```

After determining what services you need for your Web site, you can then decide how to implement those services. Using services to define the division of functionality in your Web site provides the following benefits:

® Clear and consistent development goals

By dividing your Web site into services, you enable a Web development team to easily envision the direction of development. The functionality of each service, implemented as a component, is clearly defined.

® More manageability

Because services divide the functionality of your Web site into distinct tasks, any changes in the implementation of one service will not introduce changes to another service component.

® Isolation of functionality

The functionality of a specific service is encapsulated, so any error in the implementation of a service can be easily traced to the corresponding component.

® Matching team member strengths to services

Identifying services enables you to determine which member of the Web development team is best suited to build and complete the corresponding component.

In the labs for this course, you will create a Web site for the fictitious State University. The Web site includes many user, business, and data services.

This topic describes some of these services and how they are implemented.

State University Services

The user services for the State University Web site enable a student to enroll in a class, view grades for a class, and enter feedback.

The business services for the State University Web site implement business logic when adding and deleting students. An example of this business logic would be a business service that verifies a class is not full before adding a new student.

The data services for the State University Web site perform the actual update of the State University database, as determined by the business services.

To see an illustration of some of the services used for the State University Web site, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

State University Application Design

The user services for the Web site are implemented by the user's Web browser with Hypertext Markup Language (HTML) files and Active Server Pages. Any script contained in the Active Server Pages runs on the Web server.

The business services for the State University Web site are provided by ActiveX server components installed on the Web server.

The data services are provided by Microsoft SQL Server or a Microsoft Access database. The database is installed on the State University Web server, but could have easily been installed on a separate computer.

User Services

The following table lists the files used to implement some of the user services for the State University Web site.

Service	Files
Changes mascot behavior.	mascot.htm, mascot.class, spin.ocx
Enters class information.	classview.htm, transfer.htm
Jumps to other Web pages.	links.htm
Enters Web site feedback.	feedback.htm

Business Services

The following table lists the files used to implement some of the business services for the State University Web site.

Service	Files
Adds a student to a class.	addclass.asp, add.dll
Transfers a student from one class to another.	transfer.asp, transfer.dll, add.dll, drop.dll
Drops a student from a class.	dropclass.asp, drop.dll

Data Services

The following table lists the files used to implement some of the data services for the State University Web site.

Service	Files

Modifies enrollment information.	SQL Server 6.5 Enrollment table
Stores feedback information.	SQL Server 6.5 Feedback table

To see an illustration of the physical implementation for some of the State University Web site services, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

State University Site Map

After identifying the necessary services for a Web site, and mapping those services to a hardware implementation, you can then determine how a user will navigate within your Web site. To do this, you create a site map, which is a document that shows how files are connected.

To see an illustration of the State University site map, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Web site development is best accomplished by a Web site development team. To enable the development team to build a Web site quickly and efficiently, each member should have clearly defined roles and responsibilities.

To see an animation that discusses the roles of Web site development team members and the tools used by each, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

Microsoft Visual InterDev is the product that contains the tools most team members will use for Web site development. Visual InterDev enables team members to collaborate on the development of a Web site.

This section includes the following topics:

® Web Developer

® Programmer

® HTML Author

The Web Developer is responsible for creating the Web site and writing any client-side and server-side scripts that are necessary to extend the functionality of the Web pages.

Responsibilities

The Web Developer has the following responsibilities:

® Construct Web site architecture

The Web Developer defines and constructs the Web site architecture. This includes defining pages and links.

® Invoke components

The Web Developer adds server-side script to Active Server Pages to invoke the components and controls created by the Programmer.

® Write script expressions

The Web Developer writes any client-side or server-side script that is necessary to provide specific functionality for the Web site.

Tools

The tools that enable Web Developers to complete their responsibilities include:

® Visual InterDev

The Web Developer uses Visual InterDev to define and construct the Web site architecture, and to edit HTML pages and Active Server Pages.

To see an illustration of Visual InterDev, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

® Script Wizard

Visual InterDev includes the Script Wizard, which helps the Web Developer create client-side scripts.

To see an illustration of the Script Wizard, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

The Web Developer for State University

As the Web Developer for the State University Web site, you will create a variety of files, each with a different function. The following table describes these files.

File	Description
getfeedback.asp	Uses server-side script to store feedback submitted by a student into the Feedback table in the StateU database.
profile.asp	Uses client-side script to validate user input. Uses server-side script to save the user input so other Web pages can access it.
addclass.asp	Uses server-side script to call an Add component that adds a student to a class.
dropclass.asp	Uses server-side script to call a Drop component to drop a student from a class.
transferclass.asp	Uses server-side script to call a Transfer component that transfers a student from one class to another.
class_descriptions.asp	Uses server-side script to display a list of all classes and majors.
transcript.asp	Uses server-side script to retrieve a student's transcript.
classList.asp, classForm.asp,	Uses server-side script to display classes in List

classAction.asp	view and Form view.
classview.htm	Uses controls to display class information in a list box.
mascot.htm	Uses client-side script to control a Java applet with an ActiveX control.
home.asp	Uses an **Image** control and pop-up menu control to display class information.

The Programmer is responsible for creating and maintaining the applications used for a Web site.

Responsibilities

The Programmer has the following responsibilities:

® Create ActiveX server components

The Programmer creates ActiveX server components to provide business services for a Web site.

® Create ActiveX controls

The Programmer creates ActiveX controls that run on a workstation to extend the functionality of a Web page.

® Create Java applets

The Programmer creates Java applets that run on a workstation to extend the functionality of a Web page.

Tools

The Programmer typically uses the following tools:

® Programming tools

The Programmer uses Microsoft Visual Basic 5.0 and Microsoft Visual C++ 5.0 to create ActiveX server components and ActiveX controls. The Programmer also uses Microsoft Visual J++ to create Java applets.

To see an illustration of Visual Basic 5.0, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

® Microsoft Transaction Server

The Programmer uses Microsoft Transaction Server to provide transaction and resource management for ActiveX Server components.

To see an illustration of Microsoft Transaction Server, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

The Programmer for State University

As the Programmer for the State University Web site, you will create ActiveX server components to extend the functionality of the Web site. You will also use these ActiveX server components with Microsoft Transaction Server.

The following table describes the files you will create.

File	Description
stateu.dll	Performs the enrollment functions **Add**, **Drop**, and **Transfer**. You will create stateu.dll first, but later split it into add.dll, drop.dll, and transfer.dll.
add.dll	Adds a student to a class.
drop.dll	Drops a student from a class.
transfer.dll	Transfers a student from one class to another class.

To help customers design and develop enterprise applications, Microsoft developed the Microsoft Solutions Framework. This framework is a set of guidelines that you can use when designing Web sites.

The Microsoft Solutions Framework is a collection of resources, including enterprise architecture planning and a process for designing complex, distributed applications.

The Microsoft Solutions Framework also introduces an application model that includes standards and guidelines for designing distributed, multi-tier client/server applications.

In this section, you will learn about the advantages of a service-based application model. As you work through this course, you will learn how these services are implemented, and how to build the sample State University Web site.

This section includes the following topics:

```
<%'This code creates a connection object.
Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionTimeout = Session("StateU_ConnectionTimeout")
conn.CommandTimeout = Session("StateU_CommandTimeout")
conn.Open Session("StateU_ConnectionString"), _
            Session("StateU_RuntimeUserName"), _
            Session("StateU_RuntimePassword")
%>
```

```
<%
'This code retrieves the student transcript
set rsTranscript = Server.CreateObject("ADODB.RecordSet")
rsTranscript.ActiveConnection = conn
rsTranscript.Open "select enrollment.classid" & _
   ",title,grade from enrollment,classes " & _
   "where classes.classid = enrollment.classid" & _
   " AND studentid=" & frmStudentID
%>
```

```
<%
Do Until rsTranscript.EOF
   response.write "<TD>"&rsTranscript.fields("ClassID").value&"</TD>"
   response.write "<TD>"&rsTranscript.fields("Title").value&"</TD>"
   select case rsTranscript.fields("Grade")
      case 0
         grade = "F"
      case 1
         grade = "D"
      case 2
         grade = "C"
      case 3
         grade = "B"
      case 4
         grade = "A"
      case else
         grade = "In Progress"
   end select
   response.write "<TD>"&grade&"</TD>"
   response.write "</TR><TR bgcolor=white>"
   rsTranscript.MoveNext
Loop
%>
```

```
'-- Retrieves parameters from form
<% frmResponse = Request("response") %>
<% frmEase = Request("ease") %>
<% frmInteractive = Request("interactive") %>
<% frmUseful = Request("useful") %>
```

```
<%
'-- Establish Connection to Database
Set conn = Server.CreateObject("ADODB.Connection")
conn.ConnectionTimeout = Session("StateU_ConnectionTimeout")
conn.CommandTimeout = Session("StateU_CommandTimeout")
conn.Open Session("StateU_ConnectionString"), _
          Session("StateU_RuntimeUserName"), _
          Session("StateU_RuntimePassword")
%>
```

You can use the **AdvancedDataSpace** object provided by the Advanced Data Connector to create an instance of a custom business object. A business object is an ActiveX Server component. It typically resides on the Web server.

You can create your own custom business objects using a language such as Visual Basic. For example, in the State University scenario, you can create a business object to enroll students in a class. From an .asp file, you can create an instance of the object and invoke the methods provided by the object to accomplish business functions.

In addition to creating instances of custom business objects, you can create an instance of the **AdvancedDataFactory** object, which is provided by the Advanced Data Connector. You can use the **AdvancedDataFactory** object to retrieve a set of records and access the data in the recordset programmatically. This object is automatically called by the **AdvancedDataControl**.

Creating an Instance of a Custom Business Object

To create an instance of a business object, you insert the **AdvancedDataSpace** object into a Web page and then call the **CreateObject** method of the **AdvancedDataSpace** object.

The **CreateObject** method creates an instance of the business object on the Web server and returns an object reference. You use the object reference to invoke methods of the business object.

The syntax of the **CreateObject** method is as follows. For a description of each argument, click the argument.

AdvancedDataSpace.**CreateObject** *ProgID*, *ServerName*

This example code uses the **CreateObject** function to create an instance of the **StateU.Enroll** business object using DCOM, and then it invokes the **Add** method of the object to enroll a student in a class.

```
set objEnroll = ADS.CreateObject("StateU.Enroll","myserver")
objEnroll.Add (classID, stuID)
```

Creating an Instance of the AdvancedDataFactory

In most cases, you will not need to use the **AdvancedDataFactory** directly. When you use the **AdvancedDataControl**, it invokes the **AdvancedDataFactory** for you.

However, when you use the **AdvancedDataControl** object to create a recordset, you cannot write script that reads the data in the recordset programmatically. If you need to read the data in a recordset programmatically, you can use **AdvancedDataSpace** and **AdvancedDataFactory** objects to create the recordset.

You will need to programmatically read data in a recordset if you want to display the data from the recordset in controls that are not data-aware. In that case, you must write script to read the recordset and then set the value of the controls.

u **To create a recordset using the AdvancedDataFactory**

1. Insert the **AdvancedDataSpace** control on your Web page.
2. Create an instance of the **AdvancedDataFactory** object by invoking the **CreateObject** method of the **AdvancedDataSpace** object.
3. Create a recordset by invoking the **Query** method of the **AdvancedDataFactory** object.
4. Assign data from the current record to HTML text box controls.

The following example code creates an **AdvancedDataFactory** object, queries the State University database for a list of all students, and then assigns the first and last names from the current record to HTML controls.
{ewc mvimg, mvimage,!tip.bmp}

```
set ADF = ADS.CreateObject("AdvancedDataFactory","http://myserver")
set myRS = ADF.Query("DSN=stateu;UID=sa;PWD=;","select * from students")
txtFirstName.value = myRS.fields("First_Name")
txtLastName.value = myRS.fields("Last_Name")
```

If you also want to display the data from the recordset in data-aware controls, you can insert the **AdvancedDataControl** in your Web page and add the data-aware controls. Then, you set the **Recordset** property of the **AdvancedDataControl** object to the recordset object returned by a **AdvancedDataFactory**. The following example code sets the **Recordset** property of the **AdvancedDataControl** to the recordset variable **myRs**.

```
set ADC.Recordset = myRS
```

Some data-bound controls enable users to scroll through records. Depending on the purpose of your Web page, you may want to update other controls in response to a user moving through the recordset. You can set the **Bookmark** property of a **Recordset** object to reposition the current record.

The following example code shows a data-bound list box that displays a list of student IDs. When a user selects a student ID in the list box, the script sets the **Bookmark** of the recordset to the selected item from the list box and updates the HTML text boxes.

```
Sub DBList_Click()
   myRS.BookMark = DBList.selecteditem
   txtFirstName.value = myRS.fields("First_Name")
   txtLastName.value = myRS.fields("Last_Name")
End Sub
```

To see a demonstration of how to uses the **AdvancedDataSpace** control to create an instance of the **AdvancedDataFactory** object, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

A string identifying a server-side business object that implements the rules and methods for the client application.

A string that identifies the Web server where an instance of the server-side business object is created.

You can use the Advanced Data Connector to create Web pages that run dynamic queries without downloading a new Web page from the Web server.

In this exercise, you will create a Web page that contains the **AdvancedDataControl** object and the data-bound list control **DBList**. You will insert the **DBList** control and **AdvancedDataControl** object, and then bind the **DBList** control to the **AdvancedDataControl** object.

You will add client-side script to the Web page so that when a user clicks the **ClassID List** button, the **DBList** control will display the ClassID column from the Classes table. When a user clicks the **Title List** button, the **DBList** control will display the Title column from the Classes table.

To see an illustration of how the completed Web page will look, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

u Set up ADC server components on your Web server

1. To set up the Advanced Data Connector (ADC) server components on your Web server, click this icon.
   {ewc mvimg, mvimage,!exec.bmp}

   This executable file creates a folder in C:\program files\common files\system.
2. Copy the folder msadc from C:\program files\common files\system to the WWW folder on your Web server. Typically, the WWW folder is C:\inetpub\wwwroot.

   Copying the folder creates a new Web site on your Web server. The new Web site contains sample Web pages that use the ADC controls.

u Copy files to the StateU Web project

1. Copy the file msadc10.cab from C:\program files\common files\system\msadc to the controls folder in the StateU Web project.
2. Copy the file dblist32.ocx from \MWD\Labs\Lab07.2 to the controls folder of the StateU Web project.
3. Add the file classview.htm from the folder \MWD\Labs\Lab07.2 to the root of the StateU Web project.

u Set up ADC client components on your development computer

® To install ADC client components, you go to the sample ADC Web page on your Web server. In Microsoft Internet Explorer, go to http://*servername*/msadc/samples/adctest.asp.

When you go to the sample Web page, the ADC client components and data-bound controls are installed on your development computer.

Note If your development computer is the same as the one running your Web server, you still need to go to the sample Web page to install the client components.

---

u Insert the AdvancedDataControl object and DBList control

1. Open the file classview.htm in the Visual InterDev Source Editor. Locate the comment that tells you where to insert the **AdvancedDataControl** object, and then insert the object.
   a. In the Visual InterDev Object Editor, accept all of the default properties for the control, and then close the Object Editor.
   b. Delete the DATA attribute that is inserted with the object. This will enable you to add <PARAM> tags.
   c. Set the ID attribute to ADC.
   d. Set the CODEBASE attribute to controls/msadc10.cab.
2. Create <PARAM> tags with the following NAME and VALUE attributes.

NAME	VALUE
Bindings	DBList
Server	The name of your Web server.
Connect	The connection string that connects to the StateU database.
SQL	"SELECT * FROM Classes"

3. Locate the comment that tells you where to insert the **DBList** control, and then insert the control. The name of the control in the **Insert ActiveX Control** dialog box is **Microsoft DBList Control, version 5.0**.
   a. In the Visual InterDev Object Editor, accept all of the default properties for the control, and then close the Object Editor.
   b. Set the ID attribute to DBList.
   c. Set the WIDTH attribute to 200 and the HEIGHT attribute to 168.
   d. Set the CODEBASE attribute to controls/dblist32.ocx.

Note The **DBList** control is a licensed control. If you plan to enable this control for Internet Component Download, you must create a License Package File (.lpk) for the control and add the <OBJECT> tag for the License Control Manager to classview.htm. For information about using licensed controls, see Using Licensed Controls in Chapter 4: Using Objects on Web Pages.

---

4. Create <PARAM> tags with the following NAME and VALUE attributes.

NAME	VALUE
ListField	Title
BoundColumn	Title

To see an illustration of how your HTML tags should look, click this icon.
{ewc mvimg, mvimage,!code.bmp}

u **Change the displayed list**

1. In the <SCRIPT> section of the Web page, write a procedure named Title_list that sets the **ListField** property of the **DBList** control to Title.
2. Write a procedure named Classid_list that sets the **ListField** property of the **DBList** control to ClassID.

   To see an illustration of how your script should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}
3. Save the file classview.htm.

u **Test the Web page**

1. In the browser, preview the Web page classview.htm.
2. Verify that the **AdvancedDataControl** object and **DBList** controls are working properly by checking to see if the titles of the classes are displayed in the **DBList** control.
3. Click the **Title List** and **ClassID List** buttons to see if the **DBList** control displays the appropriate ClassID or Title column from the Classes table..

In this exercise, you will modify the file classview.htm (created in the previous exercise) to use the **AdvancedDataSpace** and **AdvancedDataFactory** objects.

To see an illustration of how your completed Web page should look, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

When a student selects a class from the list box, the fields ClassID, Title, StartDate, and Seats will be displayed in four text boxes on the right side of the list box. Two command buttons will be updated, according to which field is displayed in the list box.

You will create a separate **Recordset** object by using the **AdvancedDataSpace** control and the **AdvancedDataFactory** control. To display the data in the **DBList** control, you will assign the **Recordset** object to the **AdvancedDataControl**. You will then write script to print the currently selected record in the **DBList** control to the text boxes.

For information about using these controls, see Using the Advanced Data Space in this chapter.

u Insert the AdvancedDataSpace control

1. In Visual InterDev, open the file classview.htm and find the <OBJECT> tag for the **AdvancedDataControl** object. Insert the **AdvancedDataSpace** control after the **AdvancedDataControl** object.
2. With the Visual InterDev Object Editor, accept all of the default properties for the control, and then close the Object Editor.
3. Set the ID attribute to ADS1.
4. Set the CODEBASE attribute to controls/msadc10.cab.

To see an illustration of how your HTML tag should look, click this icon.
{ewc mvimg, mvimage,!code.bmp}

u Initialize the Advanced Data Connector controls

1. In the <SCRIPT> section of the Web page, declare a global variable named myRS.

   The myRS variable will store the recordset for this Web page.
2. Create the event procedure Window_OnLoad.
   a. Create an **AdvancedDataFactory** object on the server by calling the **CreateObject** method of the **AdvancedDataSpace** object. Save the returned **AdvancedDataFactory** object in a variable named ADF1.
   b. Call the **Query** method of the ADF1 object so it will submit a query that returns all records from the Classes table. Save the returned recordset in the myRS variable
   c. Set the **RecordSet** property of the **ADC** object to the variable myRS.
   d. Refresh the **ADC** object.

   To see an illustration of how your script should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

u Write a Click event procedure for DBList

1. Create the event procedure DBList_Click. In this procedure, add code to:
   a. Move myRS to the same record on which DBList is located.

      You can do this by aligning bookmarks, as follows.

      ```
      myRS.BookMark = DBList.SelectedItem
      ```

   b. Fill in the four HTML text boxes in the HTML table by assigning the field values from the current record to the text boxes.
   c. Use the **IsNull** method to determine if any of the fields are NULL. If they are, assign an empty string ("") to the text box for the NULL field.

      The StartDate and Seats fields can have NULL values.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

u **Test the Web page**

1. Save the file classview.htm, and preview it in the browser.
2. Move through the records in the list box to verify that the text boxes contain the correct values.

```
<%
'-- Add feedback from client as new record
strSQL = "INSERT INTO feedback " & _
         "(Response, Useful, Interactive, Ease)" & _
         " VALUES (" & frmResponse & "," & frmUseful & _
         "," & frmInteractive & "," & frmEase & ")"
conn.Execute strSQL
%>
```

```
<%
'-- Add feedback from client as new record
strSQL = "INSERT INTO feedback " & _
   "(Response, Useful, Interactive, Ease)" & _
   " VALUES (" & frmResponse & "," & frmUseful & _
   "," & frmInteractive & "," & frmEase & ")"
conn.execute strSQL

'-- Check that record was added successfully
if err.number <> 0 then
   session("ErrorTitle") = "Feedback Form"
   session("ErrorText") = "The feedback could" & _
      " not be entered because of the following" & _
      " unexpected error:<p>"&err.description
   response.redirect "error.asp"
end if
%>
```

```
<!-- Transcript Table -->
<TABLE border=0 cellpadding=3>
<TR bgcolor=silver>
   <B><TD>ClassID</TD>
   <TD>Title</TD>
   <TD>Grade</TD></B>
</TR>
<TR bgcolor=white>
...
</Table>
```

```
Sub DBList_Click()
    myRS.BookMark = DBList.SelectedItem
    classid.value = myRS.fields("classid")
    title.value = myRS.fields("title")
    if IsNull(myRS.fields("StartDate")) then
        startdate.value = ""
    else
        startdate.value = myRS.fields("StartDate")
    end if
    if IsNull(myRS.fields("seats")) then
        seats.value = ""
    else
        seats.value = myRS.fields("seats")
    end if
End Sub
```

```
Sub title_list
    DBList.ListField = "title"
End Sub
Sub classid_list
    DBList.ListField = "classid"
End Sub
```

```
Sub Window_OnLoad()
    set ADF1 = ADS1.CreateObject("AdvancedDataFactory", _
        "<%http://Request.ServerVariables("SERVER_NAME")%>")
    set myRS = ADF1.Query("DSN=stateu;UID=sa;PWD=;", _
        "select * from classes")
    ADC.RecordSet = myRS
    ADC.Refresh
End Sub
```

```
<OBJECT ID="ADS1" WIDTH=19 HEIGHT=19
    CLASSID="CLSID:99586D40-DB60-11CF-9D87-00AA00B91181"
    CODEBASE="controls\masdc10.cab">
</OBJECT>
```

```
<SCRIPT LANGUAGE="VBS">
'Script code to control the AdvancedDataControl
SUB MoveFirst
    ADC.MoveFirst
END SUB

SUB MoveNext
    ADC.MoveNext
END SUB

SUB MovePrev
    ADC.MovePrevious
END SUB

SUB MoveLast
    ADC.MoveLast
END SUB

SUB Requery
    ADC.Server = Server.Value
    ADC.Connect = Connect.Value
    ADC.SQL = SQL.Value

    ADC.refresh
END SUB

SUB Init
    Server.Value = "http://<%=Request.ServerVariables("SERVER_NAME")%>"
    Connect.Value = "DSN=stateu;uid=sa;pwd=;"
    SQL.Value = "Select * from classes"
END SUB
</SCRIPT>
```

```
<!-- Insert Advanced Data Control Here -->
<OBJECT ID="ADC" WIDTH=19 HEIGHT=19
    CLASSID="CLSID:C5C18AE2-4D6E-11D0-9823-00C04FC29E30"
    CODEBASE="controls/msadc10.cab">
    <PARAM NAME="Bindings" VALUE="DBList;">
    <PARAM NAME="Server" VALUE="http://myserver">
    <PARAM NAME="SQL" VALUE="select * from classes">
    <PARAM NAME="Connect" VALUE="DSN=stateu;UID=sa;PWD=;">
</OBJECT>
.....
<!-- Insert DBList Control Here -->
<OBJECT ID="DBList" WIDTH=200 HEIGHT=168
    CLASSID="CLSID:02A69B00-081B-101B-8933-08002B2F4F5A"
    CODEBASE="controls/dblist32.ocx">
    <PARAM NAME="_ExtentX" VALUE="2646">
    <PARAM NAME="_ExtentY" VALUE="1138">
    <PARAM NAME="_Version" VALUE="327680">
    <PARAM NAME="ListField" VALUE="Title">
    <PARAM NAME="BoundColumn" VALUE="Title">
</OBJECT>
```

To customize Visual InterDev, click **Options** on the **Tools** menu.

Visual InterDev creates the project and local files in the folder C:\Program Files\DevStudio\MyProjects\StateU.

® The created files are: StateU.dsp, StateU.dsw, StateU.opt, StateU.sfl, and global.asa.

® The created folder is images.

If you accepted the default options when you installed Microsoft Internet Information Server (IIS), the files and folders created on the Web server are in the folder C:\INetPub\wwwroot\StateU.

® The created files are: global.asa and search.htm.

® The created folders are: _private, _vti_bin, _vti_cnf, _vti_pvt, _vti_txt, and images.

The <FRAME> and </FRAME> tags define a single frame in a frameset. It defines the source HTML file for the frame, as well as the look of the frame itself.

There are a number of parameters to the <FRAME> tag.

Attribute	Explanation	Example
SRC	Displays the source file for the frame. Omitting this parameter creates a blank frame.	<FRAME SRC="frame1.htm">
NAME	Provides a target name for the frame.	<FRAME NAME="top">
MARGINWIDTH	Controls the margin width for the frame in pixels.	<FRAME MARGINWIDTH="20">
MARGINHEIGHT	Controls the margin height for the frame in pixels.	<FRAME MARGINHEIGHT="10">
SCROLLING	Creates a scrolling frame. Values are Yes, No, or Auto.	<FRAME SCROLLING="Yes">
NORESIZE	Prevents the user from resizing the frame.	<FRAME NORESIZE>

In any recordset, there is one current record (unless the recordset is empty). To change the current record, you use one of the **Move** methods of the **Recordset** object.

The **BOF** (beginning of file) or **EOF** (end of file) properties are **True** if you are at the beginning or the end of the recordset, respectively. If there are no records in a recordset, both the **BOF** property and **EOF** properties are **True**. As you move through a recordset, check the value of the **BOF** and **EOF** properties to determine when you reach the end of the recordset.
{ewc mvimg, mvimage,!tip.bmp}

To see an illustration of a recordset, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

The following example code moves through all the records in a recordset.

```
Do Until rs.EOF
    'use the recordset
    rs.MoveNext
Loop
```

To retrieve data from a field in the current record, specify the field name in parentheses, as shown in the following example code.

```
rs("First_Name")
```

You can also use the **Fields** collection to loop through all fields in the current record. The following example code displays all fields from the current record.

```
For i = 0 to rs.Fields.Count -1
    Response.Write rs.fields(i)
Next
```

ADO provides a layer between your Web page and the underlying database. To work with a database, you write code that sets properties and invokes methods of ADO objects.

ADO communicates with a database driver by using OLE DB, which is a COM-based technology. OLE DB can access both SQL-based and non-SQL data. If a database provides an OLE DB provider, ADO communicates directly with the provider. If the database provides an ODBC driver, ADO communicates through MSDASQL.DLL to the driver.

To see an animation of the ADO architecture, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

The following illustration shows how ActiveX Data Objects communicate with a database.

{ewc MVIMG, MVIMAGE,!W07G015.bmp}

ADO has predefined constants that you can use to make your code easier to read and less susceptible to changes. You can provide these constant declarations by including a file in your .asp file.

If you are using Visual Basic Scripting Edition, include the file Adovbs.inc. If you are using Java Script include the file Adojavas.inc.

These files are installed with IIS on the Web server in the folder \InetPub\ASPSamp\Samples.

If you do not set the properties **ConnectionTimeOut** or **CommandTimeOut**, they will default to 15 seconds and 30 seconds, respectively.

```
'This code runs a stored procedure that requires parameters
cmd.CommandText = "GetStudentGPA"
cmd.CommandType = adCmdStoredProc
set parm = cmd.CreateParameter("StudentID",adInteger,adParamInput,4,1)
cmd.Parameters.Append parm
set parm = cmd.CreateParameter("GPA",adSingle,adParamOutput,4)
cmd.Parameters.Append parm
cmd.Execute
response.write cmd.Parameters("GPA")
```

If the **Recordset** object supports batch updating, you can cache changes to multiple records locally, and then call the **UpdateBatch** method to submit all changes to the database. Use batch updating only with a keyset or static cursor.

In Lab 4: Using Objects on Web Pages, you added a pop-up menu to home.htm and entered the list of classes in the **MenuItem** property. In this exercise, you will use the **Data Range Header** and **Data Range Footer** design-time controls to fill the pop-up menu with the correct class names for the selected major.

The following example code shows what you will add to the Click events of the three image controls:

```
Sub imgMath_Click ()
    popClasses.Clear
    <%i = 0%>
    'Insert data range header which starts a loop
        popClasses.AddItem "<%= rsClasses("title") %>",<%=i%>
        <%i = i + 1%>
    'Insert data range footer - closes loop.
    popClasses.PopUp
End Sub
```

If you do not already have a data connection to the StateU database in your Visual InterDev project, add one.

For information about adding a data connection to the StateU database, see Lab 3: Using Visual InterDev Data Tools.

u Turn on link repair for the StateU Web project

1. In FileView, right-click on the StateU project and click **Properties**.
2. On the **General** tab of the **Project Properties** dialog box, click **On** in the Link Repair section.

u Change home.htm to an Active Server Page

The script added by the data range controls is server-side script, so you need to change home.htm to an Active Server Page.

1. If you have a working copy of home.htm, release it to the Web server.
2. Rename the file to home.asp. To fix hyperlinks to the page, click **Yes**.
3. With the Visual InterDev Source Editor, open home.asp.
4. Locate the imgMath_Click event procedure.
   a. Delete the HTML comment tags (<!--...-->) that contain the script.
      You will be adding server-side script, which cannot be embedded in comment tags.
   b. In the first line of the procedure, clear the contents of the pop-up menu, as follows:
      ```
      popClasses.Clear
      ```
      This script is client-side script because it calls the method of an ActiveX control.
   c. After clearing the contents of the pop-up menu, add a line that sets up a loop counter, as follows:
      ```
      <%i = 0%>
      ```
      This script needs to be server-side script because it must run on the server as it is reading the database.

u Add the Data Range Header control

1. After the line that sets up the loop counter, insert the **Data Range Header** control.
   a. Set the ID of the control to rsClasses.
   b. Select the StateU data connection.
   c. Set the command text to **Select Title From Classes Where MajorID=1**.
      Math is MajorID 1; History is MajorID 2; Music is MajorID 3.
2. Close the Object Editor.

   For information about using the data range controls, see Using the Data Range Controls in Chapter 3: Using Visual InterDev Data Tools.

Add class names to the pop-up menu

1. After the endspan comment of the **Data Range Header** design-time control, but before the call to the **PopUp** method of the pop-up menu control, add the new data to the pop-up menu, as shown in the following code:

```
<!--METADATA TYPE="DesignerControl" endspan-->
popClasses.AddItem "<%= rsClasses("title") %>",<%=i%>
popClasses.PopUp
```

2. Increment the counter by 1, as shown in this example code:

```
<%i = i+1%>
```

Insert the Data Range Footer control

1. After incrementing the loop counter, but before the call to PopUp, insert the **Data Range Footer** design-time control.
2. Accept the default property values, and then close the Object Editor.

Save and test the page

1. Save your changes to home.asp.
2. Preview the page in the Visual InterDev InfoViewer.
3. Click the Math image to see the list of classes for the Math major.
4. Preview the page default.htm in Microsoft Internet Explorer.

   Notice that the main frame displays home.asp instead of home.htm. This is because Visual InterDev fixed the links when you renamed the file.

Repeat for History and Music

® Repeat the same steps to display the list of classes for the History and Music majors.

If you place controls on a form, you may want to validate the user-entered data before submitting the form to the Web server. Ensuring that the data is valid before sending the form to a Web server prevents unnecessary network traffic.

To validate the data entered by a user in the form fields, you can use FrontPage to add basic validation script, or you can write your own script.

Validating with FrontPage

FrontPage provides an interface that you can use to specify validation criteria for form fields. It adds client-side script that validates the data on a form before the form is submitted. If a user enters invalid data, a message will be displayed to indicate requirements for the field.

To add validation criteria with FrontPage, right-click a form field, and then click **Form Field Validation**.

To see a demonstration of how to use FrontPage to validate data in a form field, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

When you use FrontPage to add validation criteria to form fields, it creates two versions of the Web page. One version contains HTML comments that define the validation criteria. You work with this version when you edit the page with FrontPage or Visual InterDev.

The other version contains the client-side script that performs the validation of the form fields on the Web page. When a user requests the Web page, the Web server returns this version of the page.

The following example code shows the comment that FrontPage inserts into a Web page when you add validation criteria to make sure that a user fills in a text field with only letters or white space:

```
<!--webbot bot="Validation"
    s-data-type="String"
    b-allow-letters="TRUE"
    b-allow-whitespace="TRUE"
    b-value-required="TRUE" -->
    <input type="text" size="20" name="txtName">
```

FrontPage creates a function to validate the form fields and modifies the <FORM> tag to call this function in the OnSubmit event of the form.

The following example code shows the <FORM> tag modified by FrontPage:

```
<form
    action="getuserdata.asp"
    method="POST"
    onsubmit="return FrontPage_Form1_Validator(this)"
    name="FrontPage_Form1">
```

To see sample code for the validation function FrontPage_Form1_Validator(), click this icon.
{ewc mvimg, mvimage,!code.bmp}

Important When you use FrontPage to validate controls on a form, it changes the name of the form. Therefore, do not add script to the Web page that explicitly references the name of the form. You can, however, refer to the form through the **Forms** collection of the **Document** object. For example:

```
Document.Forms(0).controlname.
```

Validating with Script

If you want to add more advanced validation criteria than you can by using FrontPage, you can write your own client-side script to check the data on a form, and then submit the form or display an error message.

If you have placed a **Submit** button on a form, the form will be submitted when a user clicks the button. To

prevent the form from being submitted automatically, you can set the button type to Button rather than to Submit, and then create an OnClick event procedure for the button. You can then add script to the OnClick procedure of the button to validate data and either display an error message or use the **Submit** method of the form to submit the form.

The values in ActiveX controls and Java applets will not be submitted with a form automatically. To submit these values when a form is sent, you can add script that places the values from the ActiveX controls into hidden standard HTML controls. The values from all standard HTML controls on a form are sent automatically when the form is submitted.

The following example code shows HTML tags that define a form with one text box and a button:

```
<FORM ACTION="AddOrder.asp" METHOD=POST NAME=Form1>
    <INPUT TYPE="text" SIZE="20" NAME=Qty>
    <INPUT TYPE="button" NAME="cmdSubmit" VALUE="Submit"></p>
</FORM>

<SCRIPT Language=VBScript>
Sub cmdSubmit_Onclick ()
    If Document.Form1.Qty.Value < 5 then
        'Add code here to set hidden HTML controls
        Form1.submit
    Else
        Msgbox "Qty must be less than 5"
    End if
End Sub
</SCRIPT>
```

The OnClick event for the button checks the value in the text box. If the value is greater than 5, it submits the form.

Note There is an OnSubmit event for a form that occurs when a user clicks the **Submit** button on the form. If you have a standard button instead of a **Submit** button, and invoke the **Submit** method of the form in script, the OnSubmit event will not occur.

The **Location** object represents the URL of the current Web page. To go to another Web page, you change properties of the **Location** object.

Navigating Programmatically

To go to another location on the Web, you set the **HRef** property of the **Location** object.

The following example code goes to the Microsoft Web site:

```
Location.HRef = "http://www.microsoft.com/"
```

Running Script from a Hyperlink

You can define a hyperlink as an object, and then create an event procedure for the hyperlink that performs some logic and, depending on the result, goes to another Web page.

To define a hyperlink as an object with events, you set the HREF and ID attributes of the hyperlink. If you set the HREF attribute to a URL, the script will run, and then display the new Web page. If you set HREF to "", the script will run, but will not display a new page.

The following example code assigns the identifier JumpNext to the hyperlinked text:

```
<A HREF="" ID="JumpNext">Next Page</A>

<SCRIPT LANGUAGE=VBSCRIPT>
Sub JumpNext_OnClick()
    If Navigator.appName = "Microsoft Internet Explorer" Then
        Location.HRef = "IEPage1.htm"
    Else
        Location.HRef = "OtherPage1.htm"
    End If
End Sub
</SCRIPT>
```

In the OnClick event procedure, the script determines which browser is viewing the page, and then goes to the location for the next page on a Web site.

The primary reason for creating an event procedure for a hyperlink is to change more than one source file in response to one user event.

The following illustration shows a page with three frames. The hyperlink in the links frame has an event procedure that changes the source of all three frames on the page.

{ewc MVIMG, MVIMAGE,!W05G090.bmp}

The following example code shows this hyperlink event procedure:

```
'Change the source of the frame the hyperlink is in
Location.HRef = "Page1.htm"
'Change the source of the frame named "main"
Parent.main.Location.HRef = "Page2.htm"
'Change the source of the frame named "image"
Parent.image.Location.HRef = "Page3.htm"
```

Microsoft Internet Explorer and other browsers provide an HTML object model. This object model enables you to manipulate the browser and identify the controls contained on the forms or frames of a Web page.

The following illustration shows the HTML object model provided by Microsoft Internet Explorer. This object model is compatible with the Netscape Navigator HTML object model.

{ewc MVIMG, MVIMAGE,!W05G200.bmp}

Referring to Controls on a Form

If an object is in a form, you navigate the object model to refer to the object in client-side script. For example, to refer to the controls on a form, you refer to the document, the form, and then the control.
{ewc mvimg, mvimage,!tip.bmp}

The following example code sets the value of a text box on a form by using both the form name and the **Forms** collection.

```
<FORM NAME = Myform>
    <INPUT TYPE="TEXT" NAME=txtName>
</FORM>

<SCRIPT LANGUAGE=VBScript>
Document.MyForm.txtName.Value = "Leslie"
Document.Forms(0).txtName.Value = "Leslie"
</SCRIPT>
```

Note To refer to objects in a form directly, without navigating the object model, you can add <SCRIPT> tags within the <FORM> tag.

Referring to Controls Not on a Form

If the object is placed outside of a <FORM> tag, you can reference it directly, as shown in the following example code:

```
Calendar.Value = "September"
```

Referring to Java Applets

To refer to Java applets in script, you must use the HTML object model. The following example code uses the **Document** object to refer to the Outline Java applet on a Web page:

```
Document.outline
```

Referring to Controls in an HTML Layout

To refer to controls in an HTML Layout, you need to include the name of the HTML Layout control when referencing the control on the layout, as shown in the following example code:

```
layoutID.Calendar.Value = "September"
```

For more information about using the HTML object model, see Using the HTML Object Model.

In this section, you will learn how to add error-handling code to script, and how to use the Microsoft Script Debugger to debug your scripts.

To create a robust Web application, you should anticipate possible script errors and include error-handling code in your Web pages. The error-handling code should attempt to resolve the error or return an appropriate message to the user.

This section includes the following topics:

If a client-side script error occurs and error-handling code has not been included, Microsoft Internet Explorer will display an error message to the user in a dialog box, and will stop running the script.

To see an illustration of the **Script Error** dialog box, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

On Error Statement

You can change this default error-handling behavior for a Web page by adding the **On Error Resume Next** statement to the beginning of your script. This statement instructs Microsoft Internet Explorer to continue running the script. For each procedure in which you want error handling enabled, you must include the **On Error Resume Next** statement.

If an error does occur, Microsoft Internet Explorer will store the error information in the **Err** object. If another error occurs, the **Err** object will be updated with the new error, and the old error will be overwritten.

To turn off error handling, you use the statement **On Error GoTo 0**. Microsoft Internet Explorer will revert to the default error-handling behavior, displaying the error message to the user.

Note VBScript does not support the **On Error GoTo <*label*>** statement, and you cannot write an error handler that is called automatically when an error occurs. Therefore, you must implement inline error handling to check for an error after each statement that can cause an error.

Err Object

To detect run-time errors, check the **Number** property of the **Err** object after each statement that might cause an error. If **Number** is zero, an error has not occurred. If it is not zero, an error has occurred.

To retrieve information about the error, check the **Description** property.

When an error occurs, the **Err** object will contain the error information until another error occurs. If a statement runs successfully, the **Err** object will not be cleared. Therefore, after an error occurs, you should clear the error by invoking the **Clear** method of the **Err** object.

To see a code sample that shows the general syntax for handling errors, click this icon.
{ewc mvimg, mvimage,!code.bmp}

To test your own error-handling code, you can purposely cause an error by using the **Raise** method of the **Err** object.

VBScript does not use all available numbers for its errors. If you want to generate your own errors, begin a numbering scheme with 65535 and work your way down. For example:

```
Err.Raise 65000
```

Microsoft Script Debugger is a free add-on to Microsoft Internet Explorer 3.0. The Script Debugger provides debugging features for Web applications, and works with both JScript and VBScript.

To see a demonstration of how to use the Script Debugger, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

The following illustration shows the Script Debugger in break mode.

Some of the features of the Script Debugger include:

® Dynamic view

The debugger provides a dynamic view of your Web page. Frameset relationships are shown in a hierarchy.

® Breakpoints

You can set breakpoints anywhere in your code. In break mode, you can single-step through the code. An Immediate widow will display the value of variables.

® Call Stack

A Call Stack window displays which procedures that have been invoked.

® Syntax coloring

The HTML and script syntax is displayed with different colors to help you read and debug your script.

Note The Script Debugger opens a copy of a Web page in a temporary Internet cache. Any edits you make while running the Script Debugger will apply only to the cached Web page. To correct an error permanently, you must edit the source file on the Web server.

For more information about the Microsoft Script Debugger and how to install it, go to the Microsoft Script Debugger Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

To install the Script Debugger, see Microsoft Script Debugger in the Library included on this CD-ROM.

Visual Basic Scripting Edition (VBScript) is a subset of Visual Basic and is case insensitive. The VBScript interpreter is fast, portable, and can be freely licensed from Microsoft.

The following example code shows how VBScript declares a variable, a general procedure, and an event procedure:

```
<SCRIPT LANGUAGE=VBScript>
Option Explicit
'Script-level variable declaration
Dim id

'Sub procedure
Sub GetID()
    id = InputBox ("Enter your id number")
End Sub

'Event procedure called when user clicks button named cmdTest
Sub cmdTest_OnClick()
    GetID
End Sub
</SCRIPT>

<INPUT TYPE=button NAME=cmdTest>
```

Some of the differences between Visual Basic and VBScript are the supported data types, the scope of data, and how constants are created and used.

To see a list of Visual Basic features that are not supported by VBScript, see the VBScript folder in the Visual InterDev InfoViewer.

Data Types

VBScript supports only the **Variant** data type, which can hold different types of data. In general, you can store the data you need in a **Variant** data type, and the data will function appropriately.

To determine the type of data currently in the variable, you can use the **VarType** function.
{ewc mvimg, mvimage,!tip.bmp}

To declare a variable in VBScript, you use the **Dim** statement. The following example code declares the variable **myName**:

```
Dim myName
```

VBScript also supports arrays. All arrays in VBScript are zero-based. The first index number in an array is always zero.

The following example code declares an array that can store 11 values:

```
Dim Students(10)
```

Constants

To create a constant in VBScript, you use the **Const** statement. You can create string or numeric constants with meaningful names, and then assign literal values to them.

The following example code uses the **Const** statement to declare two constants and assigns values to them:

```
Const MYSTRING = "This is my string"
Const MYAGE
MYAGE = 37
```

Note In VBScript, you can assign a value for a constant when it is declared, or in a separate statement. To differentiate constants from variables, use all uppercase letters when creating constants.

Scope of Data

Scope determines the place in a Web page from which a variable can be referenced. The two levels of scope used in VBScript are local scope and script-level scope.

- If you declare a variable in a procedure, it has local scope. The variable is only available to the procedure. When the procedure terminates, the variable goes out of scope and loses its value.
- If you declare a variable outside of a procedure, it is available to all procedures on the Web page. This is referred to as script-level scope. A script-level variable maintains its value while the Web page is displayed in the browser.

In the following example code, the variable X is available to all procedures on the Web page, while Y is available only within the procedure MySub.

```
<SCRIPT>
Dim X           ' Script-level variable

Sub MySub()
    Dim Y       ' Local variable
End Sub
</SCRIPT>
```

Flow Control

You can use all of the Visual Basic flow control statements, such as **Do...While**, **Do...Until**, **If...Then...Else**, **For...Next**, and **Select Case**.

Built-In Visual Basic Functions

VBScript supports some, but not all, of the built-in Visual Basic functions, such as **Msgbox**, **Date**, and **IsNumeric**.

However, because VBScript is a subset of Visual Basic, not all built-in functions are supported. For example, VBScript does not support the **Format** function and the file I/O functions.

JavaScript is a C-based language developed by Netscape Communications Corporation.

The following example code shows how to declare a variable and a procedure in JavaScript:

```
<SCRIPT Language=JavaScript>
var id
function getid () {
    id = prompt ("Enter your id number");
}
</SCRIPT>

<INPUT TYPE=button NAME=cmdTest OnClick="getid()">
```

The HTML tag for the **cmdTest** button specifies the procedure to run when a user clicks the button.

JavaScript Syntax

The following list highlights some of the syntax requirements for using JavaScript.

® JavaScript is case sensitive.

® The **Var** statement declares a variable, such as:

```
var id = 5;
```

® A semicolon ends a JavaScript statement, such as:

```
id = 5;
```

® Square brackets refer to arrays, such as:

```
A[1,1] = "test";
```

® Braces {...} group statements together, such as:

```
if (qty == 5) then
{
    total = current + qty;
    tax = .08;
}
```

® All procedures are functions.

The **function** keyword must be lowercase, and you must use parentheses to invoke the function, as shown in the following example code:

```
function getid () {
    id = prompt ("Enter your id number");
}
getid ();
```

® When assigning an object to a variable, you do not use the **Set** statement. In the following example code, the object **txtID** is assigned to the variable **id**.

```
var id;
id = Document.frmControl.txtID;
```

For more information about using JavaScript, see the JScript folder in the Visual InterDev InfoViewer.

In this section, you will learn how to use the HTML object model. For complete information about the Microsoft Internet Explorer object model for scripting, go to the Microsoft Object Model Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

You can use the objects provided by the HTML object model to refer to objects on a Web page and to change the appearance and behavior of the browser.

To see an illustration of the HTML object model, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

This section includes the following topics:

The **Window** object has a **Frames** property that represents a collection of frames in a window. Each frame is also a **Window** object with its own properties, including a **Document** property that returns a **Document** object.

Scope of Script in Frames

The scope of scripting code is at the frame level of a Web page. If you want to write script in one frame to access a control in another frame, you must navigate the HTML object model to retrieve the parent window, and then use the **Frames** collection to retrieve the frame you want to access.

For example, the following illustration shows a Web page with three frames. Each frame includes a control. The command button in one frame contains script that reads data from the controls in the other frames.

{ewc MVIMG, MVIMAGE,!W05G210.bmp}

To see a demonstration of how a user interacts with this Web page, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

The files for this demonstration are included with the *Mastering Web Site Development* CD-ROM, in the folder C:\Demo Code\Ch05\Frames.

Accessing Controls in Frames

To access a different frame with script, refer to the **Frames** collection of the parent window of the current frame by using one of these syntax methods:

Parent.**Frames**("*FrameName*")

Parent.*FrameName*

To access a control in a different frame, you refer to the control name. If the control is contained in a form, you use this syntax:

Parent.*FrameName*.**Document**.*FormName*.*ControlName*

If the control is not in a form, you refer to the name of the control with this syntax:

Parent.*FrameName*.*ControlName*

In the following example code, the text box in Frame 1 is in a form, but the text box in Frame 2 is not:

```
Sub cmdRead_OnClick()
    'Read the text box value from Frame 1
    value1 = Parent.Frame1.Document.Forms(0).Text1.Value

    'Read the text box value from Frame 2
    value2 = Parent.Frame2.Text1.Value
End Sub
```

The values of both controls are read in the OnClick event procedure of the command button in Frame 0.

For more information about creating frames and assigning names to them, see Using Frames in Chapter 2: Developing a Web Project.

State University has created the interactive Java applet of their mascot by exposing methods and properties.

The following table lists the properties and methods of the Java applet mascot.class.

Property or method	Description
Stop	Method that stops the applet.
Start	Method that starts the applet.
Speed	Property that sets the time the applet pauses between flip cycles. The smaller the number, the shorter the pauses.
Direction	Property that sets the direction the flips: 1 is right to left, and 2 is left to right.

In this exercise, you will add event procedures to the controls on the page mascot.htm to access properties and invoke methods of the mascot Java applet as the user clicks the controls.

The following illustration shows the controls on the mascot Web page.

{ewc MVIMG, MVIMAGE,!W05G300.bmp}

Add an event procedure for the Stop button

1. Open mascot.htm with the Visual InterDev Source Editor.
2. Add the NAME attribute to the <APPLET> tag, and set it to mascot.
3. Add a <SCRIPT> section to the <HEAD> section of the page, and set the LANGUAGE attribute to VBScript.
4. To stop and start the Java applet, create an OnClick event procedure for the cmdStop button in the <SCRIPT> section.
   a. If the value of the button is Stop, call the **Stop** method of the Java applet, and then change the value of the button to Start.
   {ewc mvimg, mvimage,!tip.bmp}
   b. If the value of the button is Start, call the **Start** method of the Java applet, and then change the value of the button to Stop.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

   For information about creating an event procedure, see Writing Event Procedures.
5. Save and test your script for the button control.

Add an event procedure for the Change Speed button

1. In the <SCRIPT> section, create an OnClick event procedure for the cmdChangeSpeed button.
   a. If the value in the txtSpeed text box is numeric, set the **Speed** property of the Java applet to the value of the txtSpeed text box.
   {ewc mvimg, mvimage,!tip.bmp}
   b. If the value in the txtSpeed text box is not numeric, display a message box, and reset the value of the txtSpeed control to the current speed of the Java applet.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}
2. Save and test your script for the button control.

   To make the applet tumble faster, set the **Speed** property to a smaller number.

Add event procedures for the Spin button

1. Create a SpinUp event procedure for the spnSpeed button.
   a. Increment the **Speed** property of the Java applet by 10.
   b. Set the value in the txtSpeed text box to the value of the **Speed** property.

2. Create a SpinDown event procedure for the spnSpeed button.
   a. Decrement the **Speed** property of the Java applet by 10.
   b. Set the value of the txtSpeed text box to the value of the **Speed** property.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}
3. Save and test your script for the button control.

u **Add an event procedure for the Reverse button**

1. Create an OnClick event procedure for the cmdReverse button to set the **Direction** property of the Java applet.
   a. If the **Direction** property is 1, set it to 2.
   b. If the **Direction** property is 2, set it to 1.
2. Save and test your script for the controls.

To see an illustration of how your code should look, click this icon.
{ewc mvimg, mvimage,!code.bmp}

In this exercise, you will create a form that gathers profile information from students. You will write validation code for the form, and then submit the form data to an .asp file on the Web server.

In Chapter 6: Using Active Server Pages, you will learn how to write the Active Server Page that retrieves information from the form.

The following illustration shows the User Profile form. To view information about the attribute used for each control, click the control.

{ewc MVIMG, MVIMAGE,!W05G030.SHG}

Submit data from a form to an Active Server Page

1. In Visual InterDev, copy the files profile.htm and getuserdata.asp from the folder \Labs\Lab05 to the StateU Web project.
2. Open the User Profile page profile.htm with the Visual InterDev Source Editor.
3. Locate the <FORM> tag, and then set the following attributes:
   a. Set the NAME attribute to frmProfile.
   b. Set the ACTION attribute to getuserdata.asp.

      The file getuserdata.asp is an Active Server Page that echoes back the values of the controls it receives.
   c. Set the METHOD attribute to Post.

   To see an illustration of how your HTML should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

   For information about setting form attributes, see Creating HTML Forms in Chapter 2: Developing a Web Project.
4. Save your changes to profile.htm.
5. Test the form by viewing the page profile.htm in Microsoft Internet Explorer.
   a. Type an ID, a name, and a major in the controls on the page.
   b. Click the **Submit** button.

   The data will be sent to the Web server.

Note The value of the Majors combo box control will not be sent to the Web server because it is an ActiveX control, and not a standard HTML control.

Add initialization code

1. In the file profile.htm, create a <SCRIPT> section in the <HEAD> section of the page, and set the LANGUAGE attribute to VBScript.
2. Create an OnLoad event procedure for the **Window** object.
3. Fill in the lstMajors combo box with the three majors at State University, as shown in the following example code:

```
Document.frmProfile.lstMajors.AddItem "Math"
Document.frmProfile.lstMajors.AddItem "Music"
Document.frmProfile.lstMajors.AddItem "History"
```

   For information about writing page initialization script, see Initializing an HTML Page.
4. Save your changes to profile.htm, and test the page.

   You should now have data in the combo box.

Add validation code for the form

1. Change the Submit button to a Push button by setting the TYPE attribute to button.
2. Create an event procedure for the button that performs the following tasks:
   a. Validates that the ID data is a number between 1 and 57.

Note The State University database initially contains 57 students. In a production Web site, you would validate that the ID data is a valid student ID in the StateU database.

b. If the ID is invalid, displays a message and does not submit the data.

c. If the ID is valid, submits the data by calling the **Submit** method of the form.

To see an illustration of how your code should look, click this icon.
{ewc mvimg, mvimage,!code.bmp}

For information about validating controls on a form, see Validating Form Data.

3. Save your changes to profile.htm.
4. Test the form by previewing the page profile.htm in the Visual InterDev InfoViewer.

u **Add an HTML hidden control for the Majors combo box**

1. To hold the value entered in the combo box, add an HTML hidden control at the bottom of the form.

   a. Set TYPE to Hidden.

   b. Set NAME to Major.

   c. Set VALUE to an empty string, as follows:

   ```
   <INPUT TYPE=HIDDEN NAME=Major Value="">
   ```

2. Add script to the OnClick event procedure of the **Submit** button. Set the **Value** property of the hidden control to the **Text** property of the lstMajors control.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

3. Save your changes to the page profile.htm.
4. Test the page by previewing it in the Visual InterDev InfoViewer.

   a. Select an item in the Majors list box, or type a new major.

   b. Click the **Submit** button.

   The data in the combo box will be sent to the Web server.

This is a standard HTML text box with the NAME attribute set to txtID.

This is a standard HTML text box with the NAME attribute set to txtName.

This is an ActiveX combo box with the ID attribute set to lstMajors.

This is a standard HTML Submit button with the NAME attribute set to Submit.

When you create frames for a Web page, a common task is to have one hyperlink change the contents of two or more frames. In this exercise, you will edit the file links.htm to change both the main frame and the image frame when a user selects a hyperlink.

Add an event procedure to a hyperlink

1. In Visual InterDev, copy the file mascotlogo.htm from the folder \Labs\Lab05 to the StateU Web project.
2. Open links.htm with the Visual InterDev Source Editor.
3. Edit the hyperlink to mascot.htm by making these changes:
   a. Add the ID attribute, and set it to MascotLink.
   b. Set the HREF attribute to "".

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}
4. Create an OnClick event procedure for the hyperlink.
   a. Set the main frame to point to mascot.htm.
   b. Set the image frame to point to mascotlogo.htm.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

   For information about creating an event procedure for a hyperlink, see Location Object.
5. Save your changes to links.htm.
6. Test your changes by opening default.htm in the Visual InterDev browser and clicking the image of the State University mascot.

Web browsers currently provide interpreters for two scripting languages: VBScript and JavaScript.

VBScript

Visual Basic Scripting Edition (VBScript) is a subset of Microsoft Visual Basic for Applications. Microsoft Internet Explorer version 3.0 and later provides a VBScript interpreter.

Netscape Navigator version 2.0 does not provide this interpreter, however, you can acquire a Netscape plug-in to support VBScript. For information about this plug-in, go to the NCompass ScriptActive Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

For information about VBScript, go to the Microsoft VBScript Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

You can also refer to the VBScript Reference folder in the Visual InterDev InfoViewer.

JavaScript

JavaScript is a C-like language that was created by Netscape Communications Corporation. Netscape Navigator version 2.0 and later provides a JavaScript interpreter. Microsoft Internet Explorer version 3.0 and later provides a JScript interpreter, the Microsoft implementation of JavaScript.

For information about the Microsoft implementation of JavaScript, see the JScript Reference folder in the Visual InterDev InfoViewer.

For information about JavaScript, go to Netscape's JavaScript Authoring Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

Choosing a Scripting Language

VBScript and JavaScript are very similar languages. In both languages, you can define variables, create procedures, and access properties and methods of objects.

Only subtle variations in syntax define the difference between the two languages. Neither language is compiled, and both will run on any hardware platform. Both languages are interpreted, so speed differences are a result of the browser, and not of the language itself.

When choosing a scripting language, consider the following issues:

® Browser compatibility

Your user's Web browser must include a scripting interpreter for the scripting language you choose. Microsoft Internet Explorer version 3.0 and later has interpreters for both VBScript and JavaScript. Netscape Navigator version 2.0 provides an interpreter for JavaScript. You can acquire a Netscape plug-in to support VBScript.

® Programmer familiarity

You should choose a scripting language that you are most familiar with. If you have Visual Basic experience, you can quickly learn VBScript. If you have Java or C experience, JavaScript will be more familiar to you.

You can enforce explicit variable declaration by using the **Option Explicit** statement. This statement should be the first statement inside your first <SCRIPT> tag.

The **Document** object represents the Web page that is currently displayed in the browser. In the hierarchy, it is located below the **Window** object.

The **Document** object can contain multiple forms, hyperlinks, and frames. You can use this object to change the appearance of a Web page and to refer to other objects on the page, such as the **Form** and **Link** objects.

Document Properties

To retrieve information or change the appearance of the Web page, you access the **Document** object properties. To change the appearance of the document, your script must run during page initialization.

For information about initializing a page, see Initializing an HTML Page.

The following example code changes the background color of a page, and retrieves the title of the document and the date it was last modified:

```
Document.bgColor = "Blue"
txtTitle = Document.Title
Date1 = Document.lastModified
```

Document Methods

As your Web page is loading in the browser, you can add content to the page dynamically by calling the **Write** and **WriteLn** methods of the **Document** object. Both of these methods will write text to the HTML page, but the **WriteLn** method will also write a line feed.

These methods change the content of the Web page, so you must place calls to the methods in page initialization script. The position of the text that is added to a page will be based on the location of the script in the Web page.

The following example code inserts the string "Good Morning" or "Good Afternoon" in the Web page at the location of the script:

```
<SCRIPT LANGUAGE="VBScript">
If Hour(now) < 12 then
    Document.Write "<b>Good Morning!</b>"
Else
    Document.Write "<b>Good Afternoon!</b>"
End If
</SCRIPT>
```

The **Window** object represents the browser window of a Web page. It is the top-level object in the HTML object model hierarchy. You can use methods and properties of the **Window** object to modify the appearance of a window and retrieve information about the browser.

Changing Status Text

To change the text displayed in the status bar of the browser, you set the **Status** property of the **Window** object.

The following example code sets the status to Complete:

```
Window.Status = "Complete"
```

Because the **Window** object is the top-level object of a Web page, you do not need to specify it when you use methods of properties of the object, as shown in the following example code:

```
Status = "Complete"
```

Navigating Programmatically

To navigate to another page programmatically, you use the **Navigate** method of the **Window** object.

The following example code changes the current page to the home page of the Microsoft Web site:

```
Navigate "http://www.microsoft.com/"
```

Initialization and Clean-Up

The **Window** object has an OnLoad event that is called to initialize a page when the browser has finished loading the page. The object also has an OnUnLoad event that is called when the page has been unloaded.

For information about writing initialization code, see Initializing an HTML Page.

Displaying and Retrieving Information

To display and retrieve information from a user, you use the **Alert** and **Prompt** methods of the **Window** object. The **Alert** method displays a message. The **Prompt** method prompts a user for input.

The following example code prompts the user for a name with the **Prompt** method, and displays the name to the user with the **Alert** method:

```
<SCRIPT Language=VBSCRIPT>
Sub cmdTest_OnClick
    user = prompt ( "Enter your name")
    alert "hello " & user
End Sub
</SCRIPT>
```

You can refer to a form by name or by index number.

In the following example code, the form is accessed by index number:

```
Document.Forms(0).txtName.Value="Leslie"
```

1. **Assume you have an object defined in your Web page as follows:**

```
<INPUT TYPE="BUTTON" NAME="ValidateOrder" VALUE="Order">
```

When a user clicks the button, you want to display a Validating Order message box. What do you need to do?

{ewc mvimg, mvimage,!answer.bmp}

A. In a <SCRIPT> section of your Web page, create a procedure named ValidateOrder_OnClick, and add the appropriate code.

{ewc mvimg, mvimage,!answer.bmp}

B. In a <SCRIPT> section of your Web page, create a procedure named Order_OnClick, and add the appropriate code.

{ewc mvimg, mvimage,!answer.bmp}

C. Before you close the <INPUT> tag, create a procedure named OnClick, and add the appropriate code.

{ewc mvimg, mvimage,!answer.bmp}

D. In an <EVENTS> section of your Web page, create a procedure named Button_Click, and add the appropriate code.

2. **Client-side script is:**

{ewc mvimg, mvima

A. Parsed by the script interpreter on the user's computer.

ge,!
ans
wer
.bm
p}

{ew
c
mvi
mg.
mvi
ma
ge,!
ans
wer
.bm
p}

B. Compiled to intermediate code, and then run on the user's computer.

{ew
c
mvi
mg.
mvi
ma
ge,!
ans
wer
.bm
p}

C. Parsed by the script interpreter on the Web server.

{ew
c
mvi
mg.
mvi
ma
ge,!
ans
wer
.bm
p}

D. Compiled to intermediate code, and then run on the Web server.

3. It is not possible to write an event procedure for which of the following objects?

{ew
c
mvi
mg.
mvi
ma
ge,!
ans
wer
.bm
p}

A. ActiveX control

{ew
c
mvi
mg.
mvi
ma
ge,!
ans

B. Java applet

wer
.bm
p}

{ew
c
mvi
mg.
mvi
ma
ge.!
ans
wer
.bm
p}

C. Hyperlink

{ew
c
mvi
mg.
mvi
ma
ge.!
ans
wer
.bm
p}

D. Standard HTML control

4. If a control is placed on a form, which syntax should be used to set a property for that control?

{ew
c
mvi
mg.
mvi
ma
ge.!
ans
wer
.bm
p}

A. Parent.*FrameName*.*ControlName*.*PropertyName*

{ew
c
mvi
mg.
mvi
ma
ge.!
ans
wer
.bm
p}

B. *ControlName*.*PropertyName*

{ew
c
mvi
mg.
mvi
ma
ge.!
ans
wer
.bm

C. Document.*FormName*.*ControlName*.*PropertyName*

p}

{ewc mvimg, mvimage,!answer.bmp}

D. You cannot refer to controls on forms.

5. The Microsoft Script Debugger can be used for all of the following except:

{ewc mvimg, mvimage,!answer.bmp}

A. Debugging server-side script.

{ewc mvimg, mvimage,!answer.bmp}

B. Debugging client-side script.

{ewc mvimg, mvimage,!answer.bmp}

C. Viewing the relationship of files that comprise frames.

{ewc mvimg, mvimage,!answer.bmp}

D. Finding run-time errors in script.

6. What types of procedures can you create with JavaScript?

{ewc mvimg, mvimage,!answer.bmp} A. Functions

{ewc mvimg, mvimage,!answer.bmp} B. Sub-procedures

{ewc mvimg, mvimage,!answer.bmp} C. Event procedures

{ewc mvimg, mvimage,!answer.bmp} D. All of the above

7. Why should you enclose client-side script in HTML comment tags (<!-- -->)?

{ewc mvimg, mvimage,!answer.bmp} A. Script-enabled browsers require that you use comment tags to distinguish script code from HTML code.

{ewc mvimg.mvimage.!answer.bmp}

B. To hide the script from users.

{ewc mvimg.mvimage.!answer.bmp}

C. To prevent browsers that do not support the <SCRIPT> tag from displaying the script in the HTML page.

{ewc mvimg.mvimage.!answer.bmp}

D. To help programmers understand the script code.

In addition to a system DSN, you can create a file DSN (.dsn), which creates a text file with the data source information rather than a registry entry.

You can then select this file when creating a data connection in your Web project. When you use a file DSN for your data connection, Visual InterDev copies the data source information from the file to a connection string in your project. In this case, your Web project does not rely on registry entries for data source information.

This is useful if you need to copy a Web application from one Web server to another. If all the connection information is included in the connection string in the project, you do not need to define a data source name on the Web server.

```
<FORM NAME="frmProfile" ACTION="getuserdata.asp" METHOD="POST">
```

You can change the project type later by changing the project properties.

In an Active Server Page, you can create an instance of an ActiveX Server component by using the **CreateObject** method. Once you have created an instance of a component, you can access the properties and methods associated with that component.

To see a demonstration of how to call an ActiveX Server component from an Active Server Page, click this icon. {ewc mvimg, mvimage,!democlip.bmp}

The following example code shows how to use the **CreateObject** method, and then output the return value of a method to an HTML response.

```
<% Set bc = Server.CreateObject ("MyServer.MyObject") %>
<% Response.Write bc.method( ) %>
```

When you build your ActiveX server component you should specify that all parameters be passed ByVal. This is the most efficient way to pass a parameter to an ActiveX Server component.

Note If you pass parameters ByRef, you must convert each parameter into the data type that the method expects. This is because Visual Basic Scripting Edition (VBScript) uses variants for all variable types. If you call a method of an ActiveX Server component that does not accept parameters of type variant, then you must explicitly convert the arguments to the correct type.

The following example code shows how to call the **SquareIt** method on the **Math** object. It also shows how to convert the argument to an integer data type and print the result to an HTML response.

```
<% Set mathobj = Server.CreateObject ("Math.Object") %>
<%= mathobj.SquareIt (cint(5)) %>
```

Note Once you have created an ActiveX Server component by using the **CreateObject** method, its corresponding DLL will remain loaded in memory by Internet Information Server (IIS) until it stops running. If you are testing the DLL and want to recompile it, you will not be able to recompile until the DLL is freed from memory. To force IIS to free the DLL, stop and restart the Web service in the Internet Service Manager.

For more information about calling an object from an Active Server Page, see Using ActiveX Server Components in Chapter 6.

To complete the lab exercises in this chapter, you must have the required software. For detailed information about the labs and setup for the labs, see Labs in this course.

For background information on this lab, click each of these topics:

Objectives

Prerequisites

Lab Setup

To see a demonstration of the lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

To see a diagram of how the files you edit in this lab will fit into the State University Web site, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Estimated time to complete this lab: **90 minutes**

Note There are project and solution files associated with this lab. If you installed the lab files during Setup, these files are in the folder <*Install Folder*>\Labs\Lab08 on your hard disk. If you did not install the lab files during Setup, you can find them in the \Labs\Lab08 folder of this *Mastering Web Site Development* CD-ROM.

Exercises

The following exercises provide practice working with the concepts and techniques covered in Chapter 8: Creating ActiveX Server Components.

Exercise 1: Creating the ActiveX Server Component

In this exercise, you will create an ActiveX server component that implements enrollment business processes for State University. To add a student to a class, you will create an **Add** method for the object.

Exercise 2: Adding New Business Processes

In this exercise, you will add two additional methods, **Drop** and **Transfer**, to the enrollment ActiveX server component. The **Drop** method will drop a student from a class, and the **Transfer** method will drop a student from one class and add the student to another class.

After completing this lab, you will be able to:

® Create an ActiveX server component by using Visual Basic.

® Add methods to an ActiveX server component.

® Call an ActiveX server component from an Active Server Page.

Before completing this lab you must be able to:

® Use Visual Basic version 5.0 to create a new project.

® Write server-side script in an Active Server Page.

To complete this lab, you need the following:

® Visual Basic 5.0 or later, Professional or Enterprise edition

® The Visual InterDev State University Project

® The State University Microsoft SQL or Microsoft Access database

In this exercise, you will create a new project in Visual Basic 5.0, and add the **Enrollment** class module to it. You will then compile the project into an ActiveX server component, and call the component from an Active Server Page.

You will use the classview.htm file to add and drop classes. To see an illustration of how classview.htm looks, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

When you add a class using the classview.htm file, the addclass.asp file calls the **Add** object to add a student to a class. To see an illustration of how addclass.asp looks when returned to the student, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

u Create the Visual Basic project

1. Create a new local folder called \busobjects in the root folder
2. Start Visual Basic and create a new ActiveX DLL project.

   For more information on creating Visual Basic projects, see Choosing the Type of Component.
3. In the new project, remove the class module Class1 by right-clicking the module, and then clicking **Remove**.
4. Click the Project (Project1) in the **Project** window, and in the **Properties** window, change the **Name** property to **StateU**.
5. Add a reference to ActiveX Data Objects (ADO) by clicking **References** on the **Project** menu. Select **Microsoft OLE DB ActiveX Data Objects 1.0 Library**, and then click OK.

u Add the Enrollment class module

1. Copy the file enrollment.cls from the folder \MWD\Labs\Lab08 to the folder \busobjects.
2. Add the file enrollment.cls to the StateU project by right-clicking the project in the **Project** window, clicking **Add**, and then clicking **Class Module**. Select the file enrollment.cls from the \busobject folder.

   For more information on adding class modules, see Using Class Modules.

   The **Enrollment** class module contains one public method named **Add**. View the **Add** method code to see how it executes the **Add** business process.

   Note The **Add** method creates a connection to the StateU ODBC database with the UserName and Password arguments set for a SQL Server. If you are using a Microsoft Access database, delete the UserName and Password arguments of the connection.

   ---
3. Save the project in the folder \busobjects.
4. Build the StateU.dll by clicking **Make StateU.dll** on the **File** menu.

   For more information on building components, see Compiling a Component.
5. If the State University Web site is on a different server, create a folder on that computer for the business objects, and then copy StateU.dll to the folder. Then register the .dll by using the command-line utility Regsvr32.exe.

u Call the StateU Enrollment object from an Active Server Page

1. Copy the file addclass.asp from the folder \MWD\Labs\Lab08 to the root of the StateU project.
2. Open the file addclass.asp in Visual InterDev, and find the server-side script that retrieves the lngStudentID and strClassID fields from the **Request** object.
3. After the student ID and class ID are retrieved, add script that creates the **StateU Enrollment** object.
4. Call the **Add** method and pass the lngStudentID and strClassID as arguments. Store the return value from the **Add** method in a variable named errEnrollment, which will be used by the error-checking code.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

   For more information on calling components, see Calling a Component from an Active Server Page.
5. Save your changes to addclass.asp, and test the page by opening classview.htm in the browser.

6. Log into the profile with a student ID of 1 and try adding classes.

   You should not be able to add the MT100 class, because student ID 1 is already enrolled in that class. You should be able to add other classes.

7. Use DataView to verify that the Enrollment table has been updated correctly when classes are added. You can also verify that classes are added by retrieving the transcript.asp file.

The **Reset** button automatically resets the values of standard HTML controls on a form, but you have to write the script code to reset any ActiveX controls or Java applets.

```
<!--webbot bot="GeneratedScript" preview=" " startspan -->
<script language="JavaScript">
<!--
function FrontPage_Form1_Validator(theForm)
{
    if (theForm.txtName.value == "")
    {
        alert("Please enter a value for the \"txtName\" field.");
        theForm.txtName.focus();
        return (false);
    }

    var checkOK = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz??????
ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖØÙÚÛÜÝÞßàáâãäåæçèéêëìíîïðñòóôõöøùúûüýþ \t\r\n\f";
    var checkStr = theForm.txtName.value;
    var allValid = true;
    for (i = 0;  i < checkStr.length;  i++)
    {
        ch = checkStr.charAt(i);
        for (j = 0; j < checkOK.length; j++)
            if (ch == checkOK.charAt(j))
                break;
        if (j == checkOK.length)
        {
            allValid = false;
            break;
        }
    }
    if (!allValid)
    {
        alert("Please enter only letter and whitespace characters in
the \"txtName\" field.");
        theForm.txtName.focus();
        return (false);
    }
    return (true);
}
//-->
</script>
<!--webbot bot="GeneratedScript" endspan -->
```

The operators "<>" indicate "not equal." When you are writing SQL in code, the <> operators may get interpreted as HTML tags and removed from your SQL string. In this case, you can use the **Not** operator to test for values. For example:

```
sSQL = "SELECT * FROM employees WHERE (Not EmployeeID=9)"
```

There is a broken link in the default.htm file because the file links.htm is not yet in the project. In Link View, Visual InterDev finds the broken links and highlights them in red so you can fix them.

```
<SCRIPT LANGUAGE=VBScript>
Sub cmdStop_OnClick()
    If frmControls.cmdStop.Value = "Stop" Then
        Document.mascot.stop
        frmControls.cmdStop.Value = "Start"
    Else
        Document.mascot.start
        frmControls.cmdStop.Value = "Stop"
    End If
End Sub
</SCRIPT>
```

Because the text box is in a form, use frmControls.txtSpeed to refer to the control.

```
Sub cmdChangeSpeed_OnClick()
    If IsNumeric(frmControls.txtSpeed.Value) Then
        Document.mascot.speed = frmControls.txtSpeed.Value
    Else
        MsgBox "Speed must be a numeric value."
        frmControls.txtSpeed.Value = Document.mascot.speed
    End If
End Sub
```

```
Sub spnSpeed_SpinUp()
    Document.mascot.speed = Document.mascot.speed + 10
    frmControls.txtSpeed.Value = Document.mascot.speed
End Sub

Sub spnSpeed_SpinDown()
    Document.mascot.speed = Document.mascot.speed - 10
    frmControls.txtSpeed.Value = Document.mascot.speed
End Sub
```

```
'OnClick event procedure for the Reverse button
Sub cmdReverse_OnClick()
    If Document.mascot.direction = 1 Then
        Document.mascot.direction = 2
    Else
        Document.mascot.direction = 1
    End If
End Sub
```

To refer to the Java applet in script, use Document.mascot.

When you create an enterprise Web application, you should make sure that as the site changes and grows, the application is easy to maintain and update. The following paragraphs describe some issues to consider when using ADO in your applications.

Use SQL Commands

When you create an ADO **Recordset**, the recordset is created on the server. This is referred to as a server-side cursor. This is an expensive operation because it consumes server system resources.

To update a database, you can use SQL commands, instead of creating a recordset and using methods of the **Recordset** object. For example, to insert a record into a table, you can execute an SQL Insert command rather than creating a recordset and using the **AddNew** and **Update** methods. When you use the SQL Insert command, no recordset is created.

Recordset Size

If the records returned from a query can be displayed on one Web page, you can write server-side script that creates a recordset, scrolls through the recordset, returns all the data as HTML text, and then closes the recordset. In this case, you can set the cursor type for the recordset to forward-only. This is fastest type of cursor.

If the records returned from a query do not fit on one Web page, you can provide command buttons on the Web page for the user to request another page of records. To do this, you have several choices.

- When you create the initial recordset, you can store the **Recordset** object in a session variable. When the user requests a new page of records, you use the stored **Recordset** object to retrieve the next set of records. Storing a **Recordset** object consumes server resources. Therefore, this approach may not be practical if your Web site has many concurrent users.
- Another approach is to save only the number of the current record in a session variable. In this case, you create a recordset, return one page of records, save the number of the last record returned, and then close the recordset.

  When the user requests the next page of records, you query the database again, use the saved recordset number to return the next page of records, and then close the recordset again. This approach reduces data server resources because no recordsets are kept open during a session.
- A third alternative is to use the Advanced Data Connector instead of using ADO. The Advanced Data Connector stores a recordset on the client workstation rather than on the server. For more information on using ADC, see Using the Advanced Data Connector in this chapter.

Data Updates

The best approach to working with data updates is to place the update code in business objects rather than directly in .asp files. The .asp file can then create an instance of the business object and invoke methods to perform an update. You can use ADO in Visual Basic to implement business objects.

There are several reasons to place code for data updates in business objects.

- You can create multiple business objects, each one responsible for a discrete task. Breaking an application into discrete components simplifies maintenance and testing of the application.
- Business logic should be isolated from the user interface, which is provided by the .asp file. Business logic determines how a database can be modified based on rules of the business. For example, at a university, there may be a rule that students who have a grade point average below 2.0 cannot enroll in any new classes. When you write code to add a record to an enrollment table, you must ensure the student meets the required grade point average. By placing this logic in a business object, you isolate the code. If the business rule changes, you modify the business object rather than redesign the .asp file.
- A business object can be invoked by many types of clients, such as .asp files and Visual Basic or Visual C++ applications. If you have a general business object that updates a database, many applications can use the object. Code in .asp files work only with browser clients.
- If a business object fails, the error is isolated. It will not cause the failure of the entire Web server.

For more information about creating business objects, see Chapter 8: Creating ActiveX Server Components and

Chapter 9: Using Microsoft Transaction Server.

The Advanced Data Connector provides the **AdvancedDataSpace** object to create instances of remote business objects. Remote business objects are implemented as ActiveX Server components, and typically reside on a Web server.

To create an instance of a remote business object, call the **CreateObject** method of the **AdvancedDataSpace** object. **CreateObject** creates an instance of the business object on a Web server, and creates a proxy on the client to marshal method calls to the object.

The syntax of the **CreateObject** method is as follows. For a description of each argument, click the argument.

advanceddatacontrol.**CreateObject** *ProgID*, *ServerName*

When to use AdvancedDataSpace and Active Server Pages

There are several reasons you might choose to create an instance of a business objects by using the **AdvancedDataSpace** object on the client rather than from an Active Server Page on the server.

The **AdvancedDataSpace** object allows you to call multiple business objects from a single Web page without retrieving new pages. This allows more flexibility in the design of a Web page. Active Server Pages can also call multiple business objects — but they can only get information from a submitted HTML form, which is more limiting.

Furthermore, if any business objects return a large amount of data, you can use the **AdvancedDataSpace** object to retrieve the data and cache it on the client, reducing load on the server.

However, there are often times when running business objects from Active Server Pages is better. If there are clients that cannot run ActiveX controls, you must use Active Server Pages to run the business objects.

You may also have business objects performing transactions of a secure nature. For these kinds of business objects, you do not want to expose the code to create them on a Web page with the **AdvancedDataSpace** object, where it could be misused. By placing the code in server-side script in an Active Server Page, the code will never be returned to a client.

For more information about the Advanced Data Connector, see Using the Advanced Data Connector in Chapter 7: Creating Database-Aware Web Pages.

With the Advanced Data Connector (ADC), you can create efficient Web applications. This topic summarizes some of the advantages to using the ADC to design and build applications that enable data retrieval and updates on your Web site.

Data Retrieval

The Advanced Data Connector is ideal for retrieving and displaying records from both small and large recordsets. The ADC caches records on the user's computer, which enables the user to browse all of the records in a recordset without having to retrieve additional Web pages from the Web server.

Data Updates

Your Web site will be easier to maintain and will serve a larger number of users if you isolate the data updates in business objects instead of placing the data update code directly in Web pages. You use a tool such as Visual Basic or Visual C++ to create business objects. In the business object, you define methods that update the database.

From a Web page, you can use the **AdvancedDataSpace** object to create an instance of your custom business objects and then invoke methods of the object.

Platform and Browser Compatibility

Currently, the Advanced Data Connector works only in Microsoft Internet Explorer 3.0 running on an Intel platform in Windows 95 or Windows NT 4.0.

In this section, you will learn about the basic concepts of transactions and how they are processed by Microsoft Transaction Server. You will also learn about the architecture of Microsoft Transaction Server and how it manages the processing of ActiveX server components.

A transaction is a collection of changes to data. When a transaction occurs either all of the changes are made or none of them are made. Microsoft Transaction Server (MTS) monitors and directs the processing of individual transactions to ensure that changes to data are made correctly. MTS also provides an architecture that manages the processing of related business objects within a transaction. This architecture includes process and thread management, database connection management, and connectivity support.

To hear a description of Microsoft Transaction Server, click this icon.
{ewc mvimg, mvimage,!exppov.bmp}

This section includes the following topics:

In this section, you will learn how to use the Microsoft Transaction Server Explorer to create and install transaction server components on the Microsoft Transaction Server.

The Microsoft Transaction Server Explorer is the graphical interface that you use to manage and provide transaction server components.

{ewc MVIMG, MVIMAGE,!W09G025.bmp}

This section includes the following topics:

In this section, you will learn how to add transactional support to your business objects.

You will learn how to obtain access to **Context** objects, and how to indicate that a transaction is complete or aborted. You will also learn about the importance of creating stateless objects for transactions.

This section includes the following topics:

This section provides an overview of HTTP and Active Server Pages.

When a user requests a Web page, the Web browser creates an HTTP request message and sends it to the Web server. The Web server then creates an HTTP response message and returns it to the user.

This section includes the following topics:

After you determine what the user requested in an HTTP request, you can return the appropriate information to the user by using the properties and methods of the **Response** object.

In this section, you will learn how to control the information sent to a user from an HTTP response message in your Web application.

This section includes the following topics:

Active Server Pages enables you to maintain state in a Web application. State is the ability to retain user information in a Web application. A Web application can maintain two types of state:

® Application state

Information is available to all users of a Web application.

® Session state

Information is available only to a user of a specific session.

You can also maintain state on a user's computer by using cookies.

In this section, you will learn how to maintain state in a Web application.

This section includes the following topics:

® Using Cookies

® The Session Object

® The Application Object

® Using Events in the Global.asa File

You can run ActiveX server components (formerly known as Automation servers) on a Web server in response to a user request. These components enable you to extend the functionality of an Active Server Page with any resource, such as a database, located on the Web server.

In this section, you will learn how to use ActiveX server components in your Web application.

This section includes the following topics:

The HTTP protocol sends HTTP messages over a TCP/IP connection, for example, when a user requests a Web page with a URL, such as http://domainname/path/filename.html.

During an HTTP session, the Web browser creates an HTTP request message and sends it to the Web server. In response to the request, the server creates an HTTP response message that is returned to the Web browser. The response message is interpreted by the browser as a Web page.

An HTTP Session

The following illustration shows an HTTP session and the process that occurs when a user opens an HTML document on a Web server.

{ewc MVIMG, MVIMAGE,!W06G005.bmp}

The following steps describe this process:

1. The Web browser creates a TCP/IP connection to the Web server.
2. The Web browser packages a request for a Web page from a Web server into an HTTP request message, and then sends the message to the Web server by using the TCP/IP connection. The first line of the message contains the HTTP request method. For a simple page request, the GET method is used.
3. The Web server receives the HTTP request and processes it based on the request method (such as GET) contained in the request line.
4. The server then sends back an HTTP response message. Part of the response message is a status line that contains code indicating whether the attempt to satisfy the HTTP request was successful or not.
5. When the Web browser receives the HTTP response message, the TCP/IP connection is closed, and the HTTP session terminates.

If the requested Web page contains embedded objects, such as graphic images, the Web browser makes subsequent requests for each embedded object. For example, if a Web page contains three GIF images, a background sound, and an ActiveX control, six separate HTTP sessions are required to retrieve the entire page, five for the embedded objects and one for the page itself.

Stand-alone applications are designed to retain user information between versions of the application. These types of applications are also written to retain information while the user moves from one dialog box or form to another.

In contrast, a traditional Web site does not retain user information from one page to another. This is because when an HTTP request message is sent to the Web server, the TCP/IP connection is broken.

By using Active Server Pages on a Web server, you can create a Web application that retains information about the application and the user. A Web application may include the following elements:

® The virtual root

® Global data

® Data connections

The Virtual Root

A Web application consists of all the files and folders that you have created in it or added to it. These files and folders are stored on a Web server in a file system directory configured as a virtual root.

The name of the directory and the virtual root are the same as the name of your Web application.

Global Data

Unlike a Web site that uses standard HTML pages, Active Server Pages in a Web application can save global data and make it available to all pages of the application. These pages can store information that is specific to a particular user or information to all users.

Data Connections

If you intend your Web application to use data in an ODBC database, you can connect to the database by adding a data connection to your Web application. Visual InterDev generates script within the Global.asa file to save all of the information that is necessary to connect to the database when a session of the Web application is started.

For information about data connections, see Chapter 3: Using Visual InterDev Data Tools and Chapter 7: Creating Database-Aware Web Pages.

Active Server Pages enable you to write server-side scripts that manage the interactions between the user and the Web server while your Web application is running.

With Active Server Pages, you can share information among all users of your application, store information for a specific user, retrieve information passed from the user to the server, send output to the user, and work with the properties and methods of components on the server.

There are five objects that are intrinsic to Active Server Pages. You can use these objects when creating a Web application. The following table lists these intrinsic objects.

Object	Description
Request	Retrieves the values that the user passes to the Web server during an HTTP request.
Response	Controls what information is sent to a user in the HTTP response message.
Session	Stores information about a particular user session.
Application	Shares information among all users of a Web application.
Server	Provides access to resources that reside on a Web server.

The **Response** object enables you to control the information sent to a user by the HTTP response message.

Properties and Methods of the Response Object

The **Response** object provides properties and methods that you can use when sending information to the user.

The following table lists and describes some properties of the **Response** object.

Property	Description
Buffer	Indicates whether or not a response is buffered.
Expires	Specifies the length of time for which a page cached on a browser will expire.
ExpiresAbsolute	Specifies the date and time on which a page cached on a browser will expire.

The following table lists and describes some methods of the **Response** object.

Method	Description
Clear	Clears any buffered response.
End	Stops the processing of a Web page, and returns whatever information has been processed thus far.
Flush	Sends buffered output immediately.
Redirect	Sends a redirect message to the user, causing the response message to try to connect to a different URL.
Write	Writes a variable to the current HTTP output as a string.

The **Response** object also contains the **Cookies** collection that you can use to specify the value of a cookie. For information about cookies, see Using Cookies.

For more information about the **Response** object, search for "Response Object" in Visual InterDev InfoView.

Response Object Syntax

You use the following syntax for the properties and methods of the **Response** object:

```
Response.property|method
```

In the following example code, the **Expires** property of the **Response** object is set to 0. This indicates that the content of the response message returned to the user will expire immediately.

```
<% Response.Expires = 0 %>
```

If the user returns to the Web page, its content will be refreshed and displayed by the Web server.

You use the **Write** method of the **Response** object to send information to a user from within the server-side script delimiters.

Write Method Syntax

The **Write** method adds text to the HTTP response message. For example:

```
Response.Write variant
```

The variant can be any data type (including characters, strings, and integers) that is supported by your default scripting language.

The variant cannot contain the character combination %>, which is used to denote the end of a script statement. Instead, you can use the escape sequence %\>, which the Web server will translate when it processes the script.

Note If VBScript is your default scripting language, the variant cannot be longer than 1022 characters.

Using the Write Method

The following example code uses the **Write** method within a loop to display the values of each standard HTML control on a form that is sent in an HTTP request:

```
<%For Each Item In Request.Form
    Response.Write Item
Next %>
```

In the following example code, an HTML tag is added to a Web page:

```
<% Response.Write "<TABLE WIDTH = 100%\>" %>
```

The string returned by the **Write** method cannot contain the characters %> in an HTML tag, so the escape sequence %\> is used instead.

A Web application can use information from an HTTP request when a user requests a Web page.

For example, when a user submits a form by using the POST method, the values of the controls on the form will be passed in the body of the HTTP request. A Web application can then read these values and use them to return a customized Web page to the user.

In this section, you will learn how to read and use the information provided by a user.

This section includes the following topics:

The **Request** object provides access to any information that is passed to the Web server by the HTTP request message.

Request Object Collections

The **Request** object contains five collections that you can use to extract information from an HTTP request.

The following table lists and describes these five collections.

Collection	Description
ClientCertificate	The values of the certification fields in the HTTP request.
Cookies	The values of cookies sent in the HTTP request.
Form	The values of form elements posted to the body of the HTTP request message by the form's POST method.
QueryString	The values of variables in the HTTP query string, specifically the values following the question mark (**?**) in an HTTP request.
ServerVariables	The values of predetermined Web server environment variables.

For more information about the **Request** object, search for "Request Object" in Visual InterDev InfoView.

For more information about the **ClientCertificate** object, see Using Certificates in Chapter 10: Controlling Access to a Web Site.

Using the Request Object

Each collection of the **Request** object contains variables that you use to retrieve information from an HTTP request.

In the following example code, the SERVER_NAME variable of the **ServerVariables** collection retrieves the name of the Web server:

```
Request.ServerVariables("SERVER_NAME")
```

You can use the values of these variables to create dynamic HTML to return to the user.

In the following example code, the name of the Web server is used to create a hyperlink to a Web page on the same server:

```
<A HREF="http://<%= Request.ServerVariables("SERVER_NAME")%>
/MyPage.asp">Link to MyPage</A>
```

You can also access a variable directly without specifying the name of the collection. For example:

```
Request("SERVER_NAME")
```

If you access a variable directly, the Web server will search for the variable in a collection in the following order:

1. **QueryString**
2. **Form**
3. **Cookies**
4. **ClientCertificate**
5. **ServerVariables**

If a variable with the same name exists in more than one collection, the first instance encountered will be used.

You use the **QueryString** collection of the **Request** object to extract information from the header of an HTTP request message.

When a user submits a form with the GET method, or appends parameters to a URL request, you use the **QueryString** collection to read the submitted information.

The values you read from the request are the parameters that appear after the question mark (**?**).
{ewc mvimg, mvimage,!tip.bmp}

For example, when a user clicks the **Submit** button on the following form:

{ewc MVIMG, MVIMAGE,!W06G015.bmp}

The following HTTP request is made:

```
http://name_age.asp?name=Paul&age=24&sport=Hockey&sport=Baseball
```

You can loop through all of the values in a query string to extract information passed by the user.

The following example code loops through all of the values in an HTTP request:

```
<%For Each Item In Request.QueryString
    'Display the Item
Next %>
```

If more than one value is submitted with the same value name, such as a multi-select list box on a form, you can use the index of the **QueryString** collection variable to access the individual values.

The following example code shows how to access the first and second value of a variable named sport in the **QueryString** collection:

```
Request.QueryString("sport")(0)
Request.QueryString("sport")(1)
```

You use the **Form** collection of the **Request** object to extract information from the body of an HTTP request message.
{ewc mvimg, mvimage,!tip.bmp}

The **Form** collection contains the values of each standard HTML control that has a NAME attribute. When a user submits a form with the POST method, you can read the values of the controls by using the **Form** collection.

For example, when a user completes and submits a form with this HTML:

```
<FORM ACTION="submit.asp" METHOD=POST>
Name: <INPUT TYPE=TEXT NAME="name"><P>
Favorite Color: <SELECT MULTIPLE NAME="color">
                    <OPTION>Red
                    <OPTION>Green
                    <OPTION>Blue
                </SELECT><P>
<INPUT TYPE=SUBMIT NAME="cmdSubmit" VALUE="Submit">
</FORM>
```

You can read the submitted information by using the following script in the submit.asp file:

```
Request.Form("name")
Request.Form("color")
```

You can also loop through all of the values on a form to extract information passed by the user.

The following example code loops through all of the standard HTML controls in an HTTP request:

```
<% For Each Item in Request.Form
    'Display Item
Next %>
```

If more that one value is submitted for a control on a form with the same value, such as a multi-select list box, you can use the index of the **Form** collection variable to access the individual values.

The following example code shows how to access the first and second color value selected in the color list box:

```
Request.Form("color")(0)
Request.Form("color")(1)
```

To see a demonstration of how to read the control information from a form, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Instead of sending content from the response message to the user, you can use the **Redirect** method to redirect the user to another URL.

Redirect Method Syntax

When you use the **Redirect** method of the **Response** object, you provide the URL as an argument to the method.

```
Response.Redirect URL
```

The URL specifies the absolute or relative location to which the browser is redirected.

Note If you use the **Redirect** method after information has already been sent to the user, an error message will be generated.

Using the Redirect Method

The following example code uses the **Redirect** method to display a page in high or low resolution, depending on the screen resolution of the user:

```
<%
If Request.ServerVariables("HTTP_UA_PIXELS") = "640x480" Then
    Response.Redirect "lo_res.htm"
Else
    Response.Redirect "hi_res.htm"
End If
%>
```

To see a demonstration of how to use the **Redirect** method, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Posting Values to a Form

With Active Server Pages, you can define a form that posts its input values back to the .asp file that contains the form. To do this, you break the .asp file into two parts: one part that displays the form and a second part that responds to the submitted form.

To determine whether or not a request for an Active Server Page has resulted from the form being submitted, you test to see if the HTML controls contain values. If the controls do not contain values, the user has not yet submitted the form, so you need to display a blank form that the user can complete and submit.

The following example code displays a blank form:

```
<% If IsEmpty (Request("txtName")) Then
    ' Display form
Else
    ' Form was submitted
End If %>
```

To see a demonstration of how to redirect a user to a page based on a value submitted in a form, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Cookies are a mechanism by which state can be maintained in a file on the user's computer. This file is typically stored in a folder named Cookies.
{ewc mvimg, mvimage,!tip.bmp}

A cookie is like a token for a specific page that a Web server sends to a user. The user sends the cookie back to the server during each subsequent visit to that page or number of pages.

Cookies enable information to be associated with a user. You can set and get the values of cookies by using the **Cookies** collection.

When the Web server returns an HTTP response to a user, the response message may also include a cookie. The cookie includes a description of the saved range of URLs for which that cookie is valid.

A cookie is introduced to the user by including a Set-Cookie header as part of an HTTP response. Any HTTP requests made by the user included in that range will provide a transmittal of the current value of the cookie from the user back to the server.

Creating Cookies

To set the value of cookies that your Web server sends to a user, you use the **Cookies** collection of the **Response** object. If the cookie does not already exist, **Response.Cookies** will create a new cookie on the user's computer.

The following example code creates a cookie with the city set to Redmond:

```
<% Response.Cookies("city")="Redmond" %>
```

If you want the cookie to apply to all of the pages in your Web application, you set the Path attribute of the cookie to "/". For example:

```
Response.Cookies("city").Path = "/"
```

The cookie will then be sent by the browser during each request for a page in your Web application.

You can set other attributes for cookies, such as when the cookie should expire. For example:

```
Response.Cookies("Type").Expires = "July 31, 1997"
```

Using Cookies

The browser will send cookies to the appropriate pages in your Web application. To read the value of a cookie, you use the **Cookies** collection of the **Request** object.

For example, if the HTTP request sends a cookie with the city set to Redmond, then the following example code will retrieve the value of Redmond:

```
<%= Request.Cookies("city") %>
```

For more information about cookies, search for "Cookies" in Visual InterDev InfoView, or go to the Yahoo Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

You use the **Session** object to store information that is needed for a particular user session. Variables stored in the **Session** object will not be discarded when the user goes between pages in the Web application. Instead, these variables will persist for the entire user session.

The Web server automatically creates a **Session** object when a session starts. When the session expires or is abandoned, the Web server will destroy the **Session** object.

The **Session** object differs from a cookie in that state is maintained on the server instead of on the user's computer.

A session starts during the first time a user requests an .asp file from the virtual directory of your Web application. When the session starts, the Web application generates a SessionID, and sends a response to the browser to create a cookie for the SessionID. The SessionID cookie is sent to a browser, but is not stored on the user's hard disk. Instead, the SessionID cookie is maintained in the memory of the user's computer.

Note If a user does not accept the SessionID cookie, the **Session** object will not be supported for that user.

By default, a session lasts 20 minutes. You can change this default setting by modifying the following registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\ASP\Parameters\
SessionTimeout
```

Session Object Syntax

The **Session** object has two properties and one method used with the following syntax:

```
Session.property|method
```

You use the **SessionID** property to determine the session identification of a user, and the **Timeout** property to set the timeout period for a session.

You use the **Abandon** method to destroy a **Session** object and release its resources.

Using the Session Object

The **Session** object enables you to create values that store information about a user.

For example, you can create and store the nickname and hometown values of a user with the following code:

```
<% Session("nickname") = "Nancy"
Session("hometown") = "Redmond %>
```

The following example code shows how to use the information stored in the **Session** object:

```
Hello <%= Session("nickname") %>.<BR>
How is the weather in <%= Session("hometown") %>?<BR>
```

To see a demonstration of how to use the **Session** object, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

You can use the **Application** object to share information among all users of a Web application.

A Web application starts when the first user of your application requests an .asp file from the virtual root of your Web application. The application ends when the Web server is shut down.

Application Object Syntax

The **Application** object has two methods: **Lock** and **Unlock**. These methods are used with the following syntax:

```
Application.method
```

The **Application** object can be shared by more than one user, so you should use the **Lock** and **Unlock** methods to ensure that multiple users do not change a value at the same time.

Locking and Unlocking the Application Object

All users share the same **Application** object, so it is possible that two users might attempt to modify the object simultaneously. The **Lock** and **Unlock** methods of the **Application** object prevent this possibility.

To reduce the inconvenience of a user not being able to access the **Application** object when it is necessary, try to minimize the amount of time you use the **Lock** method.

The following example code shows how to use the **Lock** and **Unlock** methods when changing the value of a hit counter used in a Web application:

```
<%
Application.Lock
Application("NumVisits") = Application("NumVisits") + 1
Application.Unlock
%>
This application has been visited
<%= Application("NumVisits") %> times.
```

Notice that the **Application** object needs to be locked only while it is being modified.

Using the Application Object

You can create and store values in the **Application** object in the same way that you store values in the **Session** object. The difference is that the information stored in the **Application** object is available from the time the first user makes an HTTP request for an Active Server Page until the Web server shuts down.

For example, if you want to display information to all users of your Web application, you can create an administrative page that enables a specified person to enter information for all users. This page would then write the information to the **Application** object, so you could use the information on the other Active Server Pages. In this way, each user could receive the same administrative information from one place.

The following example code shows how you might add information collected from an administrative page to a Web application:

```
<% Application("TodaysLecture") = Request.Form("lecture")
Application("location") = Request.Form("location")
```

The following example code shows how you might use the information specified in the preceding example code:

```
Don't miss today's lecture in room <%= Application("location") %>,
titled <%= Application("TodaysLecture") %>
```

The **Server** object provides access to resources that run on a Web server. This object also enables you to set the timeout value of server-side scripts.

Server Object Syntax

You use the **Session** object with the following syntax:

```
Server.property|method
```

You use the **ScriptTimeout** property to set or determine the amount of time a server-side script will run before it times out and returns an error to the user. To create an instance of an ActiveX server component, you use the **CreateObject** method followed by the PROGID of the component.

Using the ScriptTimeout Property

By default, a server-side script has 90 seconds to finish running. This does not include server-side script running an ActiveX server component. A server-side script will not time out while a server component is processing.

You can change the default server-side script timeout by modifying the following registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\ASP\Parameters\
ScriptTimeout
```

If your script will take longer to run than the default timeout set in the registry, you can use the **ScriptTimeOut** property to change the timeout for that Active Server Page. This enables a large script to complete without timing out.

The following example code uses the **ScriptTimeout** property of the **Server** object to increase the amount of time a script will run:

```
<% Server.ScriptTimeout = 180 'SECONDS %>
```

Using the CreateObject Method

The first step in using an ActiveX server component is to create a reference to that component. Once you have created the reference, you can call the methods of the component, or set and retrieve properties of the component.

You can also use the **CreateObject** method to create a reference to a Java class.

Note Before Active Server Page technology, you called Oleisapi.dll with the name of the component that you wanted to create. With Active Server Pages, you do not need to use the Oleisapi.dll. The **CreateObject** method of the **Server** object now replaces the call to Oleisapi.dll.

ActiveX controls run on the user's computer, while ActiveX server components reside and run on the Web server.

The following example code creates a reference to the Browser Capabilities component.

```
<% Set bc = Server.CreateObject("MSWC.BrowserType") %>
```

The Browser Capabilities component enables you to determine the capabilities of the user's browser.

How the Browser Capabilities Component Works

The Browser Capabilities component compares the browser type and version number provided in the header of the HTTP request to entries contained in the Browscap.ini file stored on the Web server.

If a match if found, the component uses the capabilities for that particular browser. If a match is not found, the component uses default capabilities in the Browscap.ini file.

Customizing the Browscap.ini File

You can declare property definitions for multiple browsers in the Browscap.ini file. You can also set default values that will be used when a browser that is not listed in the Browscap.ini file makes an HTTP request.

For each browser definition, you provide an HTTP User Agent header, along with the properties and values you want to associate with that header. For information about the format of an HTTP User Agent header, go to the HTTP Specification Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

To see a sample Browscap.ini file that lists only Netscape Navigator, Microsoft Internet Explorer 3.0, and default entries for all other browsers, click this icon.
{ewc mvimg, mvimage,!code.bmp}

In the preceding code sample of Browscap.ini, setting the parent variable enables the second browser definition to inherit properties and values from the first browser. The Microsoft Internet Explorer definition includes frames=**True**, tables=**True**, and cookies=**True**.

Using the Browser Capabilities Component

You can use the Browser Capabilities component to present Web content in a format that is appropriate for a specific browser.

In the following example code, the Browser Capabilities component is used to determine whether a browser supports ActiveX controls. If it does, an HTTP response that contains ActiveX controls will be sent.

```
<% Set objBrowser = Server.CreateObject("MSWC.BrowserType")
If objBrowser.ActiveXControls = "True" Then
    'Insert ActiveX Control here
Else
    'Handle Without Control
End If %>
```

To see a demonstration of how to use the Browser Capabilities component, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

You can use the File Access component in your Web application to create and read from any text file stored on the Web server.

With text files, you can store the state of your Web application when the Web server shuts down.

Creating and Opening Text Files

The File Access component contains the **FileSystemObject** object, which you use to open or create a text file.

To open a text file, create a **TextStream** object by using the **OpenTextFile** method of the **FileSystemObject** object.

To create a text file, create a **TextStream** object by using the **CreateTextFile** method of the **FileSystemObject** object.

The following example code creates a **TextStream** object and opens a text file:

```
' Creates a FileSystem Object
Set fsVisitors = Server.CreateObject("Scripting.FileSystemObject")
' Creates a TextStream Object and opens a text file
Set fileVisitors = fsVisitors.CreateTextFile("c:\visitors.txt", True)
```

If a text file already exists, the **CreateTextFile** method will overwrite the existing file if the overwrite argument is equal to **True**.

Reading and Writing Text

After you have created a **TextStream** object with either the **CreateTextFile** or **OpenTextFile** method, you can use the methods of the **TextStream** object to read and write text.

You can use the **ReadLine** and **WriteLine** methods of the **TextStream** object to read from and write to a text file.

The following example code sets an **Application** object value equal to the value read from a text file with the **ReadLine** method:

```
Application("visitors") = fileVisitors.ReadLine
```

To see a code sample that uses the events of the **Session** and **Application** objects to count the number of visitors to the Web site and stores that number in a text file, click this icon.
{ewc mvimg, mvimage,!code.bmp}

The preceding code sample ensures that the number of visitors to the Web site is retained even when the Web server shuts down and the Web application ends.

```
<a href="" ID="MascotLink"><img src="images/mascot.gif" border="0"></a>
```

```
Sub MascotLink_OnClick()
    Parent.main.Location.Href = "mascot.htm"
    Parent.image.Location.Href = "mascotlogo.htm"
End Sub
```

In this exercise, you will add the **Drop** and **Transfer** methods to the **StateU** business object, and call the methods of the object from an Active Server Page.

You will use the transfer.htm file to submit a form that transfers a student.. To see an illustration of how transfer.htm looks, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

When you transfer a student using the transfer.htm file, the transferclass.asp file calls the **Transfer** object to transfer a student. To see an illustration of how transferclass.asp looks when returned to the student, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Create the Drop method

1. In Visual Basic, open the StateU project, and then open the **Enrollment** class module.
2. Create a new **Public** function named **Drop**.

   The **Drop** function accepts the fields lngStudentID and strClassID as parameters, and returns an **Integer** data type. Declare the parameters as **ByVal**.

   For more information on creating methods, see Creating Methods for Classes.
3. In the **Drop** function, set the return value to 0 as a default. If 0 is returned, it indicates that no errors have occurred.
4. Add code to drop the student from a class.
   a. Add the following code to create a connection to the State University database.

   ```
   Set conn = CreateObject("ADODB.Connection")
   conn.Open "DSN=StateU;UID=sa;PWD=;"
   ```

   b. Add the following code to call the **ClassCompleted** function. This function implements a business rule that checks whether the student has already completed the class. If the student has completed the class, he or she is not allowed to drop it, in which case the return value is set to 1, and the function exits.

   ```
   If ClassCompleted(conn, lngStudentID, strClassID) Then
       Drop = 1
       Exit Function
   End If
   ```

   c. Add the following code to create a transaction and execute a SQL statement that drops the student from the class.

   ```
   strSQL = "DELETE FROM enrollment where ClassID = '" & strClassID & "' AND
   StudentID = " & lngStudentID
   conn.BeginTrans
   conn.Execute strSQL
   conn.CommitTrans
   ```

   d. Close the connection.
5. Add an error handler that rolls back the transaction by calling **RollbackTrans** and sets the return value to 1.
6. Turn on error handling at the beginning of the function.

To see an illustration of how the code for the **Drop** method should look, click this icon.
{ewc mvimg, mvimage,!code.bmp}

Create the Transfer method

1. Create a new **Public** function named **Transfer**. It accepts the fields lngStudentID, strSrcClassID, and strDstClassID as parameters, and returns an **Integer** data type. Declare the parameters as **ByVal**.
2. Turn on error handling, and set the return value to 0 as a default. If 0 is returned, it indicates that no errors have occurred.
3. Add code that transfers a student from a source class to a destination class.
   a. Add the following code to create a connection to the State University database.

```
Set conn = CreateObject("ADODB.Connection")
conn.Open "DSN=StateU;UID=sa;PWD=;"
```

b. Add the following code to call the **ClassCompleted** function. This function implements a business rule that checks whether the student has already completed the class. If the student has completed the class, he or she is not allowed to drop it, in which case the return value is set to 1, and the function exits.

```
If ClassCompleted(conn, lngStudentID, strSrcClassID) Then
    Drop = 1
    Exit Function
End If
```

c. Add the following code to create a transaction and execute SQL statements that drop the student from the source class and add the student to the destination class.

```
strDropSQL = "DELETE FROM enrollment where ClassID = '" & strSrcClassID & "'
AND StudentID = " & lngStudentID
strAddSQL = "INSERT INTO enrollment (ClassID, StudentID) VALUES ('" &
strDstClassID & "'," & lngStudentID & ")"
conn.BeginTrans
conn.Execute strDropSQL
conn.Execute strAddSQL
conn.CommitTrans
```

d. Close the connection.

4. Add an error handler that rolls back the transaction by calling **RollbackTrans** and sets the return value to 1.
5. Rebuild the StateU DLL.
6. If the Web server containing the State University Web site is on a different computer, copy the new StateU.dll to the \busobjects folder of that computer. Because the public methods have changed, you register the DLL again on that computer by using the command-line utility Regsvr32.exe.

To see an illustration of how your code for the **Transfer** method should look, click this icon.
{ewc mvimg, mvimage,!code.bmp}

u **Modify the Drop Web page**

1. In Visual InterDev, open the StateU project.
2. Copy the file dropclass.asp from the folder \MWD\Labs\Lab08 to the root of the StateU project.
3. Open the file dropclass.asp, and locate the server-side script that retrieves the lngStudentID and strClassID fields from the **Request** object.
4. After the student ID and class ID are retrieved, add script that creates the **StateU Enrollment** object.
5. Call the **Drop** method and pass the lngStudentID and strClassID as arguments. Store the return value from the **Drop** method in a variable named errEnrollment, which will be used by the error-checking code.
6. Save your changes to the file dropclass.asp.
7. In the browser, view the page classview.htm.
8. Log into the profile with a student ID of 1. You should be able to drop from all classes except for MT100, which has already been completed by student ID 1.
9. Use DataView to verify that the Enrollment table was updated correctly when classes were dropped. You can also verify that classes are dropped by retrieving the transcript.asp file.

Note If you drop a class in which the current student ID is not enrolled, the drop will still succeed. This is because the **Drop** method uses the SQL DELETE statement to remove records from the enrollment table. The SQL DELETE statement will succeed even if the record does not exist.

---

u **Modify the Transfer Web page**

1. In Visual InterDev, open the StateU project.

2. Copy the files transfer.htm and transferclass.asp from the folder \MWD\Labs\Lab08 to the root of the StateU project.
3. Open the file transferclass.asp, and locate the server-side script that retrieves the lngStudentID, strSrcClassID, and strDstClassID fields from the **Request** object.
4. After the student ID, source class ID, and destination class ID are retrieved, add script that creates the **StateU Enrollment** object.
5. Call the **Transfer** method and pass the lngStudentID, strSrcClassID and strDstClassID as arguments. Store the return value from the **Transfer** method in a variable named errEnrollment, which will be used by the error-checking code.
6. Save your changes to the file transferclass.asp.
7. In the browser, view the page transfer.htm.
8. Log into the profile with a student ID of 1 and try to transfer between classes. As student 1, you should be able to transfer from any class except MT100, which is already completed.
9. Use DataView to verify that the Enrollment table has been updated correctly. You can also verify that classes are transferred by retrieving the transcript.asp file.

When a user requests a Web page, the IIS Web server determines whether or not to return the requested Web page.

This illustration provides an overview of the security process used on each request.

{ewc MVIMG, MVIMAGE,!w10g010.bmp}

IP Address Permitted

You can grant or deny access to your entire Web server based on the Internet Protocol address of the user making the request.

u **To control access by IP address**

1. In Internet Service Manager, right-click the WWW service and click **Service Properties**.
2. On the **Advanced** tab, choose either **Granted Access** or **Denied Access**.

   To see an illustration of this dialog box, click this icon.
   {ewc mvimg, mvimage,!illust.bmp}
3. Click **Add**, and in the **IP Address** box, type the IP address of the computer to be granted or denied access, and click OK.

This security method has limitations. An IP address can change, and users can use different computers with different IP addresses. One situation in which you may want to use IP address security is to exclude everyone except known users.

User Permitted

If the IP address is valid, then IIS determines whether the login ID of the client is a valid Windows NT account.

Windows NT security requires assigned user accounts and passwords. When a user requests information from a Web server, the user is mapped to an account.

If the Web server allows anonymous logon, a user can access the Web server without providing a login ID. In this case, the user will be mapped to the anonymous logon account.

Internet Server Permissions

Next, IIS checks the Internet Server permissions for the virtual directory of the Web page that was requested. You can set Read-Only or Execute-Only permissions on virtual directories by using the Internet Service Manager.

NTFS Permissions

Finally, Windows NT determines the permissions the user has on the requested Web page. You can use Windows NT File System (NTFS) to assign permissions to Windows NT accounts for folders and files on the Web server.

Windows NT requires assigned user accounts and passwords. If you want to allow everybody to access your Web server, you must either provide a valid Windows NT account for every user or allow anonymous logon.

Anonymous logon allows users to access your Web server without providing a user ID and password. When an IIS Web server receives an anonymous request, it maps the user to a special anonymous logon account, referred to as the Internet Guest account. The user receives the access rights that have been granted to this account.

Allowing Anonymous Access

To allow anonymous access

1. In Internet Service Manager, double-click the WWW service to display its property sheet, then click the **Service** tab.
2. Click **Allow Anonymous** and click OK.

Setting the Account Used for Anonymous Access

The anonymous logon account must be a valid Windows NT user account on the Web server, and it must have the **Log on Locally** user right.

When you install IIS, it creates an account named IUSR_*computername*. For example, if the computer name is marketing, the account name will be IUSR_marketing. By default, this account is used for anonymous Internet logons.

IIS adds the IUSR_*computername* account to the Guests group and receives any permissions assigned to that group. You should review the settings for the Guests group to ensure that they are appropriate for the IUSR_*computername* account.

To change the account or password used for anonymous access

1. In Internet Service Manager, double-click the WWW service to display its property sheet, then click the **Service** tab.
2. In the **Anonymous Logon** user name text box, type an account name.

   This account must be an existing Windows NT account and must have **Log On Locally** permission.
3. In the **Password** box, type the new password.

   This password must match the password of the existing Windows NT account. If you change the password in this dialog box, be sure to change the corresponding password in the Windows NT User Manager.

There are a variety of issues involved with security on an intranet or the Internet, such as verifying the identity of users, securing communication between a client and server, and determining the author of an ActiveX control.

In general, the most important security concern is to determine who can access your Web site and what files a user can access. For example, you may want to allow all users to access most of your Web site, but restrict a few pages to certain users. If you don't set any security limitations, everybody will be able to access all of the Web pages on your Web site.

This illustration shows the architecture of a Web site, and highlights the security concerns of the client, the server, and those shared between the client and server.

{ewc MVIMG, MVIMAGE,!W10G040.bmp}

Ensure Software Accountability and Integrity

Downloading an executable component, such as an ActiveX control, can pose a security risk to your computer. A malicious ActiveX component could attempt to read or delete inappropriate files on your computer. Before allowing the component to download, you may want to determine what company created the component and confirm that it has not been altered. If you know that the control was created by a respected company, you can feel more secure in downloading the control.

Microsoft provides Authenticode technology that allows users to view the digital signature of the developer of the control. When you access a Web page that contains a signed ActiveX control, you can determine who created the control and confirm that the control has not been altered. You can then choose whether or not to download the control.

For more information on using signed controls, see Using Signed Controls in Chapter 4: Using Objects on Web Pages.

Control Access to Information and Resources

Microsoft Internet Information Server is built on the Windows NT security model. Windows NT security requires that all users use a valid Windows NT account to log on. You can either require users to provide account information, or allow them to logon with an anonymous account created by the Web server. You can control access to your Web server by limiting the rights of these accounts.

You can also use the Windows NT File System (NTFS) to assign permissions to Windows NT accounts for folders and files.

Verify Identity of Users

There are several methods you can use to confirm the identity of a Web user.

One way to verify identity is to prevent anonymous logons and require that users provide a valid Windows NT account.

Another method of identifying users is to allow anonymous logons and to write your own authentication procedures that prompt a user for an ID and password. This method requires that you maintain a database of valid IDs and passwords. Although this may work well for small applications, the management and protection of passwords can become expensive and impractical for large high-volume networks.

A third method of identifying users is to use digital certificates. When a digital certificate is installed in a Web browser, the browser automatically sends the certificate to the Web server without any user interaction. A digital certificate is issued by a Certificate Authority and confirms the identity of a user or server. Using digital certificates Internet Explorer 3.0 can authenticate servers and Internet Information Server 3.0 can authenticate clients. This method of authentication provides the highest level of security and the simplest administrative requirements. For more information on using digital certificates, see Using Certificates in this chapter.

Ensure Private Communications

When you send information between a Web server and a client computer, you can encrypt the data to ensure that other Internet users cannot view or modify it during transmission.

Microsoft Internet Explorer 3.0 and Internet Information Server support the Secure Sockets Layer (SSL) 3.0 and Private Communications Technology (PCT) 1.0 protocols for private point-to-point communication.

For information on using the SSL protocol, see Requiring Certificates in this chapter.

```
Sub Submit_OnClick()
    If IsNumeric(Document.frmProfile.txtID.Value) Then
        If Document.frmProfile.txtID.Value >= 1 AND _
           Document.frmProfile.txtID.Value <= 57 Then
            'submit the form data to the server
            Document.frmProfile.Submit
        Else
            MsgBox "ID must be between 1 and 57."
        End If
    Else
        MsgBox "ID must be numeric."
    End If
End Sub
```

```
Sub Submit_OnClick()
    If IsNumeric(Document.frmProfile.txtID.Value) Then
        If Document.frmProfile.txtID.Value >= 1 AND _
           Document.frmProfile.txtID.Value <= 57 Then

            'save the value of the combo box in the hidden control
            frmProfile.Major.Value = frmProfile.lstMajors.Text
            'submit the form data to the server
            Document.frmProfile.Submit

        Else
            MsgBox "ID must be between 1 and 57."
        End If
    Else
        MsgBox "ID must be numeric."
    End If
End Sub
```

1. Which Active Server Pages intrinsic object would you use in your Web application to extract information from an HTTP request message?

{ewc mvimg, mvimage,!answer.bmp}
A. The **Request** object

{ewc mvimg, mvimage,!answer.bmp}
B. The **Response** object

{ewc mvimg, mvimage,!answer.bmp}
C. The **Session** object

{ewc mvimg, mvimage,!answer.bmp}
D. The **Server** object

2. When is a Session object created to indicate the start of a new user session?

{ewc mvimg, mvimage,!answer.bm
A. When a user requests a Web page from a Web application.

p}

{ewc mvimg, mvimage,!answer.bmp}

B. When a user logs on to a Web server.

{ewc mvimg, mvimage,!answer.bmp}

C. When a user requests an Active Server Page from a Web application.

{ewc mvimg, mvimage,!answer.bmp}

D. When the WWW service starts.

3. Which event procedure always runs when a Web server is shut down?

{ewc mvimg, mvimage,!answer.bmp}

A. Session_OnStart

{ewc mvimg, mvimage,!answer.bmp}

B. Session_OnEnd

{ewc mvimg.mvimage.!answer.bmp}

C. Application_OnStart

{ewc mvimg.mvimage.!answer.bmp}

D. Application_OnEnd

4. Which of the following server-side script examples will not read data from an HTML form?

{ewc mvimg.mvimage.!answer.bmp}

A. Request.Form("controlname")

{ewc mvimg.mvimage.!answer.bmp}

B. Request("controlname")

{ewc mvimg.mvimage.!answer.bmp}

C. Request.QueryString("controlname")

{ewc

D. Request.ServerVariables("controlname")

{mvimg,mvimage,!answer.bmp}

5. Which of the following statements about cookies and session variables is false?

{ewc mvimg,mvimage,!answer.bmp}

A. Cookies are sent with the HTTP request; session variables are not.

{ewc mvimg,mvimage,!answer.bmp}

B. Cookies save information on the user's computer; session variables are saved on the Web server.

{ewc mvimg,mvimage,!answer.bmp}

C. Cookies can be created with server-side script; session variables cannot be created with server-side script.

{ewc mvimg,mvimage,!answer.bmp}

D. Cookies are destroyed automatically by the Web browser; session variables are destroyed by the Web server.

6. What should you do to access the properties and methods of an ActiveX server component in an Active Server Page?

{ewc mvimg.mvimage.!answer.bmp}

A. Set a reference to the component's type library, and use the **New** keyword when you declare a variable of that type.

{ewc mvimg.mvimage.!answer.bmp}

B. Install the ActiveX server component on the Web server, and create a variable with the **New** keyword.

{ewc mvimg.mvimage.!answer.bmp}

C. Install the ActiveX server component on the Web server, and use the **CreateObject** method of the **Server** object.

{ewc mvimg.mvimage.!answer.bmp}

D. Install the ActiveX server component on the user's computer, and use the <OBJECT> tag.

7. Which of the following tags sets the language for server-side script?

{ewc mvimg.mvimage.!answer.bmp}

A. <% LANGUAGE="VBScript" %>

{ewc

B. <SCRIPT LANGUAGE="VBScript">

{ewc mvimg, mvimage,!answer.bmp}

C. <LANGUAGE="VBScript">

{ewc mvimg, mvimage,!answer.bmp}

D. <%@ LANGUAGE="VBScript" %>

{ewc mvimg, mvimage,!answer.bmp}

This section covers how you can use the features of IIS to control access to your Web server.

This section includes the following topics:

One of the main security issues for Web sites is to verify the identity of a user. Typically, users provide a login ID and password as identification. Instead of requesting a login ID, a Web server can require that a user provide a digital certificate. This section shows how you can use digital certificates with Internet Explorer and IIS to verify the identity of users.

Microsoft provides a set of technologies that is used by Internet Explorer and Internet Information Server to address security issues. This set of technologies is referred to as the Microsoft Internet Security Framework (MISF). Developers can also use these technologies when creating applications that require a high level of security, such as electronic commerce. For information about MISF, go to the Microsoft Internet Security Framework Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

For a discussion of security, click this icon.
{ewc mvimg, mvimage,!exppov.bmp}

MISF contains a technology called client authentication. To use client authentication, users request a digital certificate from a Certificate Authority (CA) and install the certificate into a browser. When the user accesses a Web page that requests a certificate, the browser sends the certificate to the server, which then verifies the identity of the user. This authentication process provides a secure and straightforward means of identification and access control.

This section includes the following topics:

In Lab 5: Adding Client-Side Script, you created a form on the Web page profile.htm. In this exercise, you will set the ACTION attribute of the form to post its data back to itself.

You will save the data in the **Session** object for use by other pages of the State University Web site.

u **Rename profile.htm to profile.asp**

® In Visual InterDev, rename the file profile.htm to profile.asp.

u **Read form data**

1. In the Visual InterDev Source Editor, open the file profile.asp.
2. Change the ACTION attribute of the <FORM> tag to call profile.asp.
3. Before the <FORM> tag, add the following server-side script to test whether or not the data on the form has been filled in:

   ```
   <% If Not (IsEmpty(Request("txtID"))) Then
   ...
   <% End If %>
   <FORM ... >
   ```

   If this test is True, the page was requested by clicking the **Submit** button on the form. If it is False, the page was requested by going to a page in a Web browser.
4. If the form data is filled in:
   a. Add server-side script that reads the values of all controls sent from the form on the page profile.asp.

      For information about reading form data, see Using the Form Collection.

   b. Output the values entered in the **txtName** and **Major** controls.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}
5. Save your changes to profile.asp.
6. Test the code in profile.asp.
   a. Preview the file in the Visual InterDev InfoViewer.
   b. Enter data in the form.
   c. Click the **Submit** button.

u **Save form data in Session variables**

To make the form data available to other pages in the State University Web site, save the data in **Session** variables.

1. After saving the form data in local variables, add server-side script to profile.asp to save the form data in **Session** variables.

   For information about saving user data, see The Session Object.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}
2. Save your changes to profile.asp.

u **Use session data in another file**

1. Copy the file transcript.asp from the folder \Labs\Lab06 folder to the StateU Web project.
2. In transcript.asp, use the information stored in the **Session** object variables to output the user's Student ID in the header table for the page.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}
3. Save your changes to transcript.asp.
4. Test the **Session** object variables:

a. Preview profile.asp in the Visual InterDev InfoViewer.
b. Enter data in the form.
c. Click the **Submit** button.
d. Go to the page transcript.asp.

All users of the State University Web site should log on before visiting the pages in the Web site. In this exercise, you will route all users to the Profile page, where they will supply information about themselves.

Redirect all users to profile.asp

1. Open the global.asa file by using the Visual InterDev Source Editor.
2. If a user attempts to start the session with a different page, add code to the Session_OnStart event procedure that redirects the user to profile.asp.
   a. Save the name of the Profile page in a local variable.
   {ewc mvimg, mvimage,!tip.bmp}
   b. Save the name of the requested page in a local variable.
   The server variable SCRIPT_NAME contains the name of the page that is requested.
   c. Use the **strcomp** function to determine if the user is requesting the Profile page.
   The **strcomp** function does a case-insensitive string comparison.
   d. If the user is not requesting the Profile page, redirect the user to that page.

   For information about redirecting a user to a different Web page, see The Redirect Method.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}
3. Save your changes to global.asa.
4. Test your code by viewing any Active Server Page in the StateU Web project, other than profile.asp.

Note You need to start a new session each time you want to test the script in profile.asp. To start a new session, start a new instance of Microsoft Internet Explorer.

Redirect users to their originally desired page

Add code to the files global.asa and profile.asp to send users to the start page they requested after they entered their user information in profile.asp.

1. In global.asa, add script to the Session_OnStart event procedure that saves the name of the requested page in a session variable named requestedPage.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}
2. Save your changes to global.asa.
3. Open profile.asp with the Visual InterDev Source Editor.
4. Before the <HTML> tag, add server-side script to test whether the data on the form has been filled in.
   {ewc mvimg, mvimage,!tip.bmp}
5. If the form data is filled in:
   a. Read the values of all controls sent from the form on the page profile.asp, and save the values in session variables.
   b. Redirect users to the page they originally requested, or to default.htm if they originally requested profile.asp.
   {ewc mvimg, mvimage,!tip.bmp}

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}
6. Save your changes to profile.asp.
7. Test your changes by trying to open an Active Server Page other than profile.asp in the Visual InterDev InfoViewer.

You use the Internet Service Manager to set Read or Execute access permission on the virtual directories of your Web site.

u **To set access permissions on virtual directories of your Web site**

1. In the Internet Service Manager, double-click the WWW service to display its property sheet, then click the **Directories** tab.
2. Select the virtual directory for which you want to set permissions, and click **Edit Properties**.
3. Select the appropriate access rights, then click **Apply** to apply the changes to the Web server.

Read Permission

For a Web user to view an HTML file, the virtual directory in which it exists must have Read permission. If a user requests an HTML file that is in a directory without Read permission, the Web server returns the error*: HTTP/1.0 403 Access Forbidden (Read Access Denied - This Virtual Directory does not allow objects to be read.)*.

Note You should disable Read permission for directories containing programs and scripts such as Active Server Pages or Common Gateway Interface (CGI) applications to prevent users from downloading the applications. For information about security and .asp files, see the Knowledge Base article Active Server Pages Script Appears in Browser in the Library section.

Execute Permission

Execute permission enables a Web client to run programs and scripts on the Web server, such as Active Server Pages, CGI scripts, and Internet Database Connector (IDC) files. If a client sends a request to run a program or a script in a folder that does not have Execute permissions, the Web server returns an error.

For a Web user to access an .asp file, the virtual directory in which it exists must have Execute permissions.

A digital certificate, also known as a Digital ID, is an identifier assigned to a user or a Web server by a Certificate Authority (CA). A digital certificate provides an electronic means of proving one's identity, much like a passport does in face-to-face interactions.

There are several types of certificates, including client and server certificates. A client certificate is issued to a Web user. A server certificate identifies a Web server. A Web server must have a server certificate installed in order to request digital certificates from clients.

Certificate Authorities

Certificate Authorities (CA) are companies that verify the identity of a requester and issue digital certificates. When you accept a digital certificate as an identifier, you place your trust in the CA. You assume that the CA has done the background work to confirm the identity of the requester.

In Microsoft Internet Explorer and Microsoft Internet Information Server, you can specify which CAs you trust. In this way, only certificates from trusted CAs will be accepted.

u To specify which CAs you trust

1. In Microsoft Internet Explorer, on the **View** menu, click **Options**.
2. From the **Security** tab, click **Publishers**.
3. Select or clear a specific site.

To see an illustration of the Site Certificates dialog box in Microsoft Internet Explorer, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Classes of Certificates

There are several classes of digital certificates. Each class has a different level of security and involves a different level of verification from the CA.

For the least secure class of digital certificate, the CA may ensure only that the requester has a unique Internet e-mail address. For higher levels of security, the CA may require that a requester provide notarized public documents before the digital certificate is issued.

When a Web browser submits a certificate to a Web server, the certificate information is stored in a **ClientCertificate** collection. You can write ASP script that uses the **Request** object to retrieve information from the **ClientCertificate** collection.

Note Before you can use the **ClientCertificate** collection, you must configure your Web server for secure communication and to request client certificates. For information on how to configure your Web server, see Requiring Certificates in this chapter.

Syntax to Request Information

You use the following syntax to request information from a certificate:

Request.**ClientCertificate** (*Key*[*SubField*])

The **Key** argument is a string that indicates which piece of information you want to retrieve from the certificate. Key can have the following values:

Value	Meaning
Subject	Returns information about the subject of the certificate.
Issuer	Returns information about the issuer of the certificate.
ValidFrom	Returns a date specifying when the certificate becomes valid. This date follows VBScript format and varies with international settings. For example, in the U.S. the format is: 9/26/96 11:59:59 PM.
ValidUntil	Returns a date specifying when the certificate expires.
SerialNumber	Returns the serial number of the certificate.
Certificate	Returns a string containing the binary stream of the entire certificate.
Flags	Returns a flag that provides additional client information. The following flags are available: ceCertPresent — A client certificate is present. ceUnrecognizedIssuer — The last certification in this chain is from an unknown issuer. {ewc mvimg, mvimage,!tip.bmp}

You use the **SubField** argument with the "Subject" or "Issuer" keys to retrieve an individual field from these keys. You append a SubField value to the **Key** argument as a suffix. For example: "IssuerO" or "SubjectCN". The following table lists some common SubField values.

Value	Meaning
C	Specifies the name of the country of origin.
O	Specifies the company or organization name.
OU	Specifies the name of the organizational unit.
CN	Specifies the common name of the user. (This subfield is used with the "Subject" key.)
L	Specifies a locality.
S	Specifies a state or province.
T	Specifies the title of the person or organization.
GN	Specifies a given name.
I	Specifies a set of initials.

Requesting Information

The following example script retrieves the organization name of the subject of the client certificate.

```
<%
If (Request.ClientCertificate("SubjectO")="Msft")
   Response.Write("Good Choice!")
End if
%>
```

The following example code uses a **For...Each** statement to loop through all the keys of the **ClientCertificate** collection.

```
<%
For Each key in Request.ClientCertificate
   Response.Write( key & ": " & Request.ClientCertificate(key) & "<BR>")
Next
%>
```

If you want your IIS Web server to request digital certificates from Web users, you must enable the Secure Sockets Layer (SSL) protocol on the appropriate virtual directory of your Web site.

SSL is a security protocol that resides between TCP and application protocols such as HTTP. Because SSL depends only on TCP/IP, other client/server applications besides Internet applications can use this protocol.

SSL uses digital certificates to authenticate a client and a server. Therefore, you must use digital certificates if you want to have secure transmissions.

Once you configure the Web server to use SSL and digital certificates, a user can request a secure Web page by using the syntax https:// rather than http://. If a user uses http:// to access a virtual directory that has SSL enabled, the user receives an error message.

u **To configure your IIS Web server to use SSL and request a certificates**

1. From the IIS or Peer Web Services program group, choose Internet Service Manager.
2. Double-click the WWW service, then click the **Directories** tab.
3. Select a virtual directory, then click **Edit Properties**.
4. Click **Require Secure SSL Channel**.

   This will cause client and server certificates to be used.

To see a demonstration that shows how to access a Web site using HTTPS, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

The Microsoft Certificate Server is a Windows NT-based server product for creating and managing digital certificates. Instead of requesting a digital certificate from a CA, you can use Certificate Server to create digital certificates.

Features

Some of the key features of the Certificate Server are:

® Customizable policy

You can set the policy that you want to enforce to issue a certificate. For example, one organization may decide to grant certificates only if identification is presented in person. Another organization may automatically grant a certificate to e-mail requests.

® Key management

Because the Certificate Server uses CryptoAPI to separate key management from the server code, everything from software to smart cards and industrial grade hardware devices can be used to protect keys.

® Integrated and extensible

The Microsoft Certificate server runs as an Windows NT service and is tightly integrated with Windows NT. The certificate server exposes an application programming interface (API) and Component Object Model (COM) interfaces that can be used for customization. Developers can customize the certificates that are issued.

Usage Scenarios

Some common scenarios in which you might use Certificate Server include:

® Connecting remote employees to corporate intranets

An organization can issue digital certificates to remote employees who travel frequently. Remote employees can use the digital certificate to gain access over the Internet to secure resources on the corporate network.

® Creating a subscription-based business on the Internet

Internet businesses can issue certificates to identify and authenticate valid subscribers.

The **QueryString** collection is a parsed version of the QUERY_STRING variable of the **ServerVariables** collection. The **QueryString** collection enables you to retrieve the QUERY_STRING parameters by name.

To implement business objects, you use ActiveX server components.

To see an illustration of how State University has implemented ActiveX server components for its Web site, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

ActiveX server components offer several advantages:

- They can be created with a variety of programming languages, such as Visual Basic, Visual J++, and C++.
- They work with Microsoft Transaction Server. For information about Microsoft Transaction Server, see Chapter 9: Using Microsoft Transaction Server.
- For more information about the advantages of using ActiveX objects, see Chapter 1: Planning an Enterprise Web Site.

You can use Visual Basic to test an ActiveX DLL before using it on a Web server.

In Visual Basic, create a project group. A project group is a collection of projects. When you create a project group, you can use one project in the group to test another.

To test an ActiveX DLL

1. Open the Visual Basic ActiveX DLL project you want to test.
2. On the **File** menu, click **Add Project**, and then click **New Standard EXE**.

   This will add a new project with its own template to the Project Group window.
3. To make it the start-up project, right-click the new project, and then click **Set as Start Up.**

   Whenever you run the component, your new project will start first.
4. In the new project, add a reference to the ActiveX DLL project by clicking **References** on the **Project** menu, and then selecting the ActiveX DLL project.
5. In the project, add a command button to the form.
6. In the Click event for the command button, add code that creates an instance of a class defined in the ActiveX DLL, and then call any methods you want to use for testing the component, as shown in the following example code:

```
Dim x
Set x = New Class1
x.SomeMethod
..
```

You can trace the source code of the ActiveX DLL by setting a breakpoint on the line that invokes the method. When execution stops at the breakpoint, you can step into the source code for that method in the ActiveX DLL.

To see a demonstration of how to test an ActiveX DLL, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

For more information about testing ActiveX server components, see the technical article, Testing ActiveX Server Components in the Articles and White Papers section of the Library included on this CD-ROM.

For information about debugging and testing ActiveX components, see "Creating ActiveX Components" in the Component Tools Guide in Visual Basic Books Online.

If the form method is GET, the **QueryString** collection will contain all information passed in the form.

The following articles are from previous Mastering Series titles, or were written by the Mastering Web Site Development team to augment the course material.

® Adding Database Functions with ASP
® Browser Detection with JScript
® Building an Online Store, Part I
® Building an Online Store, Part II
® Create an Online Gallery with ActiveX Controls
® Frame Tags
® HTML Table Tags
® Packaging ActiveX Controls
® Redirecting Users to the Login Page
® Server-Side Includes
® Server Side Scripting: A Primer
® Syntax for Standard HTML Controls
® Testing ActiveX Code Components
® The <EMBED> Tag
® Using Floating Frames
® Using JavaScript to Validate a Form
® Using Session Objects to Extend the Functionality of ASP Applications
® Using Session Objects with ADO

This section contains white papers that provide in-depth discussion on technical topics related to Web Development.

The following white papers are included:

® Active Server Pages FAQ
® Active Server Pages Installation Notes
® Case Study - Residential Funding Corporation
® The Client/Server Solutions Series
® Controls that Control Controls
® Creating Your Own Visual InterDev Templates
® Creating Your Own Visual InterDev Themes
® Designing Intelligent Control Palettes
® Evaluating Web Application Development Tools
® FAQs About HTML Coding for Internet Explorer 3.0
® Internet Component Download
® Leveraging Your Visual C++ Experience on the Internet
® Licensing ActiveX Controls
® Microsoft Advanced Data Connector 1.0 FAQs
® Microsoft SQL Server in the Active Internet
® Microsoft SQL Server Scalability
® Microsoft SQL Server: An Overview of Transaction
® Microsoft Transaction Server
® Microsoft Transaction Server Installation Notes
® Microsoft Visual InterDev
® Microsoft Visual InterDev Frequently Asked Questions
® Microsoft Visual InterDev Installation Notes
® Microsoft Visual InterDev Reviewers Guide
® Scripting a Pair of Transaction Processors
® Signing Code with Microsoft's Authenticode Technology
® Unified Browsing with ActiveX Extensions
® Using Active Server Pages
® The Visual Programmer Fingers WinSock Functions from Visual Basic

The following questions and answers were compiled from the IE-HTML mailing list by Steve Pruitt. For information on subscribing to IE-HTML or other mailing lists, see Mailing Lists in the Resources Section.

This section includes FAQs on the following topics:

Steve Pruitt leads some of the Microsoft Internet Technology mailing lists, including IE-HTML and VBScript. He previously led the CompuServe sections dedicated to Windows Help, Multimedia Viewer, and other multimedia products. He also provides consulting and contract services specializing in hypertext authoring through his business, **Beyond Help LLC**. He wrote the book *Microsoft Multimedia Viewer How-To*, which is used by Viewer and MediaView authors. In his spare time, Steve is also Business and Finance Category Manager on the Microsoft Network.

How can I create a blank cell in a table that displays its borders?

Put a non-breaking space character in the cell using " ".

How can I space out a table cell with blank lines?

Users often ask how they can space out items vertically in a tabled list. Adding a <P> tag at the end of each cell doesn't work, because Internet Explorer ignores tags that don't contain any content. Instead, use <P> and
 tags (or <P> and <P> tags, depending on how much space you need) and add a non-breaking space in between:

```
<p> <br>
```

Why does my page display as blank with some browsers?

Pages that use frames must also include content for browsers that do not support frames. You include the alternate content in the <NOFRAMES> section of your frameset document. If you don't provide this content, your page will appear blank in Internet Explorer 2.0 or other browsers that don't support frames. See the next question for more information on coding for non-frame browsers.

How do I code frames to be compatible with non-frame browsers?

The document containing your frameset commands must also contain a <NOFRAMES> section that contains the HTML material you want the other browsers to display; for example:

```
<HTML>
<HEAD>
</HEAD>
<FRAMESET COLS="80,100%" FRAMEBORDER="no" FRAMESPACING=0>
<FRAME SRC=  NAME=  >
<FRAME SRC=  NAME=  >
<NOFRAMES>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#0000FF" VLINK="#000000">
Content to be displayed by other browsers...
</BODY>
</NOFRAMES>
</FRAMESET>
</HTML>
```

Browsers that don't support frames will ignore the <FRAMESET>, <FRAME>, and <NOFRAMES> tags and will display the contents of the <BODY> section.

Note The <FRAME> tags should not appear between <BODY> and </BODY> tags.

How do I create a link that changes the content of a different frame?

Include TARGET="framename" in the link; for example:

```
<A HREF="http://www.my.com/my.htm" TARGET="frame1">
```

Note that target names are case-sensitive. You can also use any of the following predefined target names:

Target name	Description
_blank	Load this link into a new, unnamed window.
_self	Load this link within the current frame.
_parent	Load this link into your parent frame (same as self if current frame has no parent).
_top	Load this link at the topmost level (same as self if current frame is at the top).

Why do my links open a new window instead of changing the frame contents?

Internet Explorer interprets target frame names strictly and is case-sensitive. For example, if you define a frame called "FRAME1" and have an anchor that links to "frame1," the target name will not be recognized. Links to undefined targets open new windows.

How do I create a link that changes two or more frames?

One easy method is to link to a new frameset page using TARGET="_top". Any documents or images common

to both framesets will be loaded quickly from the cache.

You can change two frames at once by using scripting within the anchor:

```
<A LANGUAGE = "VBScript" HREF = "http://target1.htm" TARGET = "frame1"
onClick = "parent.frame2.location.href = 'http://target2.htm'">
```

The HREF and TARGET on the first line specify the update for one frame, and the second line includes a script that changes the second frame. Substitute the appropriate frame names in each case. Note that you may need other changes in the script, depending on the structure of your page.

You can use scripting to change selected frames and (optionally) to perform other functions from a single click, as follows:

```
<A HREF="" name="ClickMe">Click me!</A>
<SCRIPT Language = "VBScript">
<!--
SUB ClickMe_OnClick()
   parent.FrameOne.location.href="my_gif.gif"
   parent.FrameTwo.location.href="my_htm.htm"
END SUB
-->
</SCRIPT>
```

How do I reference floating frames?

If you are using only floating frames, you can reference them exactly like standard frames:

```
<IFRAME SRC="page1.htm" NAME="Frame2"></IFRAME>
parent.frames(0).location.href="newpage.htm"
<A HREF="newpage.htm" TARGET="Frame2">
```

If you combine floating frames with regular frames, the floating frames have to be treated differently. Floating frames are added to the document object, not to the '"regular" frames collection. (This information wasn't included in the SDK documentation, but will be added in a future version of the documentation.)

To access a floating frame on the current page, use:

```
document.FloatingFrameName.whatever
```

To access a floating frame in a different frame, use:

```
top.frames[n].document.FloatingFrameName.whatever
```

How do I refresh a frame or an entire window during testing?

To refresh a frame, right-click within the frame and choose Refresh from the pop-up menu, or left-click within the frame and click the Refresh button on the Internet Explorer toolbar. Press F5 to refresh the entire frameset.

How do I print the contents of a frame?

To print the contents of a frame, right-click within the frame and choose Print from the pop-up menu, or left-click within the frame and click the Print button on the toolbar. You cannot print the complete multi-frame window.

How can I use an image as a frame border?

You can use an image as a frame border by adding extra "border" cells to your table. Here's an example of a table that has a 5-pixel-wide "border" tiled with the image "foo.gif":

```
<HTML>
<BODY>
```

```
<TABLE BORDER=0 CELLSPACING=0>
<!-- First row: 3 cells wide and provides a "top border" -->
<TR>
   <TD HEIGHT=5 COLSPAN=3 BACKGROUND="foo.gif"></TD>
</TR>
<TR>
   <!-- The following is the "left border" -->
   <TD WIDTH=5 BACKGROUND="foo.gif"></TD>
   <TD>
      <!-- Your table contents go here -->
      <FONT SIZE=7>Hi there!</FONT>
   </TD>
   <!-- The following is the "right border" -->
   <TD WIDTH=5 BACKGROUND="foo.gif"></TD>
</TR>
<!-- Last row: 3 cells wide and provides a "bottom border"
-->
<TR>
   <TD HEIGHT=5 COLSPAN=3 BACKGROUND="foo.gif"></TD>
</TR>
</TABLE>

</BODY>
</HTML>
```

How are measurements handled?

The handling of pixel measurements changed in the final release of Internet Explorer 3.01. In particular, the method for calculating line-height changed between the beta release and final release of the browser. If you start out by commenting out all references to "line-height" in your code (I usually do this by renaming it to "xline-height"), you will see that your content will not be visible; you will simply need to reselect your line-height values differently.

In general, you should specify font sizes and line-height using point size rather than pixels. A font that is 16 pixels tall may look fine on the screen, but it looks pretty small on a 300-dpi printer. For indents and other measurements, you might consider using inches.

What's the difference between DIV and SPAN?

Think of <DIV> as a container version of
. The
 tag allows you to add a simple line break into your document. The <DIV> tag, being a container, also allows you to specify additional attributes, such as CLASS for assigning a CSS class to this section.

If all you want to do is provide simple highlighting of a word or short phrase in a longer sentence without line breaks, use <SPAN>. This is a "do nothing" container tag that implies no default formatting of its own. However, it is a container, so you can assign a CLASS (or STYLE) attribute to it to achieve your goal.

Thus, to highlight a word in a sentence:

```
This is your last <span style="background:yellow; color:red;

font-weight:bold">Warning</span>, proceed at your own risk.
```

Are external style sheets supported in Internet Explorer 3.0?

Yes, they are supported. However, BODY settings do not work in external style sheets in the current version of Internet Explorer.

To use external style sheets, place your style definitions in a separate file with a .CSS extension, and link to this file from your Web page as follows:

```
<LINK REL=STYLESHEET TYPE="text/css" SRC="http://www.my.com/mystyle.css">
```

The external file must not include the

```
<style>
<!--
```

and

```
-->
</style>
```

tags that are used for style sheets within a document.

Why is Internet Explorer 3.01 displaying different margins than version 3.0?

If you've specified margins for multiple tags in your style sheet, you'll encounter a difference in display between Internet Explorer 3.0 and 3.01. Internet Explorer 3.0 was incorrectly handling the margin attributes (margin-left, margin-right, margin-top) as absolute values. Internet Explorer 3.01 correctly uses cumulative values if more than one style sheet specification applies.

For example, let's say you specified:

```
<BODY STYLE="margin-left: 1in; margin-top: 1in">
<P STYLE="margin-left: 0.5in; margin-top: 1.5in">Text...</P>
</BODY>
```

In Internet Explorer 3.0, a paragraph will have a 0.5-inch left margin and 1.5-inch top margin — that is, the last margins you specified (the <P> margins) will be used as the margins for the document.

In Internet Explorer 3.01, the same paragraph will have a 1.5-inch left margin and 2.5-inch top margin — that is, the margins for the <P> tag will be added to the margins for <BODY>.

To fix your documents so they display correctly in both Internet Explorer 3.0 and 3.01, collapse the two margin specifications so that only one specification applies to any text segment, as follows:

```
<BODY>
<P STYLE="margin-left: 1.5in; margin-top: 2.5in">Text...</P>
</BODY>
```

For a more detailed explanation and examples, see "Updated Margin Usage in Style Sheets" on the http://www.microsoft.com/workshop Web site.

How can I create shadowed text?

Although CSS does support shadowed text (via text placement on the page), you should be careful using this feature because non-CSS aware browsers will not be able to display it correctly. Think about how your page will look in other browsers, and determine whether the benefit of getting shadowed text via CSS is worth the cost of damaging your information in non-CSS browsers.

There are some excellent examples of text shadowing and other capabilities available via CSS on http://www.microsoft.com/truetype/css/gallery/entrance.htm, but these examples are not compatible with non-CSS browsers.

How can I set a font for the entire document?

Place a BODY command within the document head as follows:

```
<HTML>
<HEAD>
<STYLE>
<!--
BODY  {font-family:COMIC SANS MS}
-->
</STYLE>
</HEAD>
<BODY>
<H1>This</H1> is a sample page
</BODY>
</HTML>
```

The BODY command cannot be used in a linked external style sheet in the current version of Internet Explorer 3.0.

Which fonts can I use?

A font has to reside on the user's computer in order to be displayed. This means that you must use fonts that either (a) you can count on to be on the user's system or (b) users can legally download through your site. You can list multiple fonts in order of preference to handle the cases where a desired font isn't available.

The fonts shipped with various Microsoft products are listed at: http://www.microsoft.com/truetype/iexplor/popular.htm.

Be careful: Most commercial fonts are not licensed for redistribution. Read the license carefully for any fonts you have purchased before making them available on your site. Several fonts are installed with the full version of Internet Explorer 3.0, but are not included with the minimal installation. You can offer users a link to the Microsoft TrueType page (http://www.microsoft.com/truetype/) to install these fonts, or use the following links to download individual fonts:

® Verdana: ftp://ftp.microsoft.com/developr/drg/truetype/verdana.exe
® Comic Sans MS: ftp://ftp.microsoft.com/developr/drg/truetype/comic.exe
® Arial Black: ftp://ftp.microsoft.com/developr/drg/truetype/ariblk.exe
® Impact: ftp://ftp.microsoft.com/developr/drg/truetype/impact.exe

How can I control the use of the Enter key to submit a form?

If you have a single form on your page, pressing the Enter key will submit the form, even if you don't have an <INPUT TYPE=SUBMIT> button. This can bypass your validation script. The easiest way to prevent this action is to define a second form on the page. It can be empty, or it may contain only a hidden form.

For example, place this after the real form:

```
<form method=get>
<input type="hidden">
</form
```

How can I have multiple Submit buttons on my form?

Although HTML won't allow multiple Submit buttons, you can simulate this with graphics that serve as Submit buttons combined with some scripting. For example:

```
<script language="javascript">
<!--

function submit_Form()
   {
      var formVariable = document.forms[0].myLittleTextBox.value
      location.href='http://name.domain/path/ISAPIApp?myLittleTextBox=' +
formVariable
   }

function reset_Form()
   {
      alert('form being reset')
      document.forms[0].myLittleTextBox.value=''
   }
function submit_Form_DownloadFile()
   {
   //code for downloading file
   }

function submit_Form_ReadInformation()
   {
   code for reading information
   }
//-->
</script>

<FORM METHOD="POST" ACTION="javascript:submit_Form()" name="testForm">
<input type="textbox" name="myLittleTextBox">
<INPUT TYPE=IMAGE SRC="babble.gif" NAME="whatever">
</FORM>

<a href="javascript:submit_Form_DownloadFile()"><IMG SRC="babble.gif"></a>
<a href="javascript:sumit_Form_ReadInformation()"><IMG SRC="babble.gif"></a>
<a href="javascript:reset_Form"><IMG SRC="babble.gif"></a>
```

How can I put a Submit button in a different frame from the form?

You can't have the form's Submit button in another frame, because the button has to exist within the same <FORM></FORM> containment as the elements it needs to submit. However, you can call the Submit method of the form from another frame. For example:

```
<input type=button name=remote value="submit">
<script language="vbscript">
<!--
Sub remote_OnClick
   parent.document.frames(1).document.forms(0).submit
end sub
-->
</script>
```

This code will put a Submit button in the first frame of the page. When the user clicks the button, it will call the Submit method of the first form in the second frame.

How can I send mail from my forms?

Internet Explorer handles a **mailto** command by opening a new mail message with the address filled in, but it does not insert the Subject line or message text as some other browsers do. Internet Explorer doesn't use an internal mail system; it can work with many different mail systems. These systems use inconsistent interfaces, so Internet Explorer can't determine how to pass the other information in a mail message.

The most dependable way to use e-mail with a form is through a script or program on the server. Many ISPs provide such scripts to anyone with a Web site on their servers.

Warning If you use a script that sends mail from your account to an address specified in the HTML page, someone could capture your HTML source and modify it to cause your server to send mail from your account to any address they want. For security, always use server programs or scripts that have the destination address hard-coded or that only work with HTML pages located on that server.

How do I create a marquee that scrolls vertically?

You need to use the ActiveX marquee control that is included with Internet Explorer 3.0. The CODEBASE parameters are not required because this control will always be installed with the browser. The page that is to display the marquee will include code similar to the following (see the next question for specifics on how to control the scrolling direction and other variations):

```
<OBJECT
   ALIGN=CENTER
   CLASSID="clsid:1a4da620-6217-11cf-be62-0080c72edd2d"
   WIDTH=100 HEIGHT=90 BORDER=0 HSPACE=0
   ID=marquee >
<PARAM NAME="ScrollStyleX" VALUE="Circular">
<PARAM NAME="ScrollStyleY" VALUE="Circular">
<PARAM NAME="szURL" VALUE="marqcont.htm">
<PARAM NAME="ScrollDelay" VALUE=60>
<PARAM NAME="LoopsX" VALUE=-1>
<PARAM NAME="LoopsY" VALUE=-1>
<PARAM NAME="ScrollPixelsX" VALUE=0>
<PARAM NAME="ScrollPixelsY" VALUE=-3>
<PARAM NAME="DrawImmediately" VALUE=0>
<PARAM NAME="Whitespace" VALUE=0>
<PARAM NAME="PageFlippingOn" VALUE=0>
<PARAM NAME="Zoom" VALUE=100>
<PARAM NAME="WidthOfPage" VALUE=100>
</OBJECT>
```

The page that *is* the marquee (referred to marqcont.htm in the code above) will have code similar to the following:

```
<TABLE BORDER=0>
 <TR>
   <TD WIDTH=100 HEIGHT=60 ALIGN=CENTER VALIGN=CENTER>
     <SPAN STYLE="font: 10pt/12pt Arial; color: purple; font-weight: bold">
     This site is enhanced for
     <BR>
     <IMG ALIGN=CENTER width=88 height=31 SRC="ie_stat.gif" border=0>
   </TD>
 </TR>
</TABLE>
```

What other scrolling effects can I create with marquees?

The *ScrollPixelsX* and *ScrollPixelsY* values control the direction of scrolling movement. X values control horizontal movement, and Y values control vertical movement. Positive values move to the right or bottom, negative values move to the left or top. Thus, X=0 and Y=-3 results in scrolling from bottom to top. If both the X and Y values are non-zero, the movements are combined, resulting in a slanted movement.

The *Zoom* parameter can be used to cause an explosive effect by contracting and expanding the contents.

Including the following lines in the OBJECT definition causes the contents of the marquee to switch between different contents when the user right-clicks the marquee.

```
<PARAM NAME="szURL" VALUE="alphabet.htm">
<PARAM NAME="PageFlippingOn" VALUE="1">
<PARAM NAME="OtherURL0" VALUE="alphabet2.htm">
<PARAM NAME="OtherURL1" VALUE="alphabet3.htm">
```

How can I control the background color of a marquee?

You cannot use BackColor with the Marquee control. Instead, set the BGCOLOR in the body of the HTML file being displayed by the control.

Why is my marquee being formatted incorrectly?

You cannot use any other tags, such as FONT or anchor, inside MARQUEE tags. They can be used outside, applying to the entire marquee, as follows:

```
<A HREF="nextpage.htm">< <FONT SIZE=+2><MARQUEE>Message Here</MARQUEE></FONT></A>
```

See the Frames section for multi-browser questions relating to frames.

How can I detect which browser is being used?

Whenever possible, you should be testing for the capabilities of the browser (for example, VBScript support) rather than identifying the browser. This will automatically handle upgraded versions of browsers. If that's not practical, you must use scripting code such as the following.

```
<SCRIPT LANGUAGE="JavaScript">
<!--
var ver = navigator.appVersion;
if (ver.indexOf("MSIE") != -1)
{
 window.location.href="ie.htm"
}else
   window.open("netscape.html", target="_self")

// --></SCRIPT>
```

How can I embed a sound for both Internet Explorer and Netscape Navigator?

You can now use the EMBED tag for both browsers, as follows:

```
<EMBED SRC="sound.au" height=2 width=2 autostart=true hidden=true>
```

If you're going to use the sound as part of a complex page, place the tag towards the bottom of your code (but still within </BODY>). Wave files can take a little time to load and tend to hold up the display of your page while they do so. (This may cause users to hit the stop button in frustration before they see what you have to offer.)

If you have problems with MIDI files, be sure your server has the proper MIME types for MIDI files:

MIME Type	Description	Suffix/file extension
audio/x-midi	MIDI	.MID
audio/midi	MIDI	.MID

How can I prevent the colors in my page from being dithered?

You should only use the 216 colors in the standard palette supported by both Internet Explorer and Netscape Navigator to ensure that your page will look good on systems running in 256-color mode. These colors include any combination of red, green, and blue components of 0, 51, 102, 153, 204, and 255.

How can I resize a control when the window is resized?

Place the control in a frame of the desired size, with WIDTH="100%" and HEIGHT="100%".

Why does a dot appear after an image?

A dot appears after an image if there is white space after the image tags in an anchor. This can come from a space like this:

```
<A HREF="some.htm"><IMG SOURCE="some.gif"> </A>
```

It will also happen if the anchor is split across two lines, like this:

```
<A HREF="some.long.URL.htm" TARGET="_top"><IMG SRC="some.gif">
</A>
```

How can I get rid of the gray box that displays while loading the Layout control?

Simply place a WIDTH=0 inside the HTML Layout Control Object tag:

```
<OBJECT CLASSID="CLSID:812AE312-8B8E-11CF-93C8-00AA00C08FDF"
ID="c" width=0 STYLE="LEFT:0;TOP:0">
</OBJECT>
```

How can I turn off the toolbar and menu bar? How can I display in kiosk mode?

There are several ways to accomplish this — choose the one that best fits your needs. Be sure to give the user a way to leave your site, unless you have a custom application that runs only under controlled conditions.

You can open a new window with the desired options:

```
<SCRIPT LANGUAGE="VBScript">
<!--
window.open "http://www.microsoft.com/", "window2", "toolbar=yes,
location=yes, directories=yes, status=yes, menubar=yes, scrollbars=yes,
resizeable=yes, width=300, height=250"
-->
```

You can switch the current window to and from kiosk mode:

```
<script language="vbscript">
<!--
Sub window_onLoad()
dim exp
set exp=Explorer
exp.fullscreen=true
End Sub
Sub window_onUnload()
dim exp
set exp=Explorer
exp.fullscreen=false
end sub
-->
</script>
```

You can start Internet Explorer in kiosk mode with the -k switch:

```
iexplore -k http://www.mysite.com
iexplore -k file://d:\cfo\html\cfomain.htm
```

What screen resolution should I design for?

Surveys indicate that the great majority of users operate in 640x480 resolution with 256 colors. However, you should always try to find out the requirements of your target audience and base your design on that data. For example, if you are trying to sell high-end technology to power users, you can count on different requirements.

You should always view your site in the widest possible range of screen resolutions and color depths. If there is a problem under some configurations, you will need to fix it before your users discover it. If you are running Windows 95, be sure to install the QuickRes PowerToy so you can change resolutions and color depths easily. The PowerToys are at http://www.microsoft.com/windows/software/powertoy.htm. Also check out http://www.microsoft.com/windows/software.htm for other valuable free software.

How do I redirect a user to a new URL?

The following example will display a line of text for three seconds before automatically jumping to a new URL. Note that the quotes around the CONTENTS value must surround both the time delay and the target URL. The delay can be zero, but remember that the page will still be visible until the new page is downloaded.

```
<HTML>
<HEAD>
<TITLE>Old page</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF TEXT=#000040>
<META HTTP-EQUIV="REFRESH" CONTENT="3; URL=newpage.htm">
<FONT FACE="ARIAL" SIZE=3>
<i>We've moved!  I'll take you to our new page now...</i>
<BR>
</FONT>
</BODY>
</HTML>
```

Why don't my local links work in Internet Explorer 3.0?

Names that contain multiple words separated by spaces will not work in Internet Explorer. These names are not compliant with W3C specifications, although some other browsers accept them.

Why don't my links to a section within a page work?

The NAME value is case-sensitive. You must use the exact same form when it's defined and when it's referenced. These will work:

```
<A NAME="Section3">
<A HREF="#Section3">
<A HREF="page2#Section3">
```

These won't work:

```
<A NAME="Section3">
<A HREF="#section3">
<A HREF="page2#section3">
```

How can I describe a link in the status bar when the mouse points to an image?

Include a simple JScript in the anchor, like this:

```
<a href="main.html" target="main" onMouseOver="window.status='Back to the first
page'; return true"><img src="home.gif" alt="home" border=0
hspace=0 vspace=0 ></a><br>
```

How can I keep my page from being cached?

To keep your page from being cached (e.g., if your content is dynamic and determined by a server application), use the META tag as follows:

```
<META HTTP-EQUIV="Expires" CONTENT="Tue, 04 Dec 1993 21:29:02 GMT">
```

Note that the expiration date should be already past. META tags are used in the <HEAD> section of the HTML page. Refer to http://www.w3.org/pub/WWW/Protocols/HTTP/1.1/spec.html for further details. Internet Explorer does not support HTTP-EQUIV Pragma no-cache.

How can I display small animations in my page?

There are currently two ways to display animations: using AVI files and using animated GIFs.

- AVIs are movies that can include sound and more than 256 colors. AVI files can be compressed using a number of techniques. AVIs are currently only supported by Internet Explorer.
- Animated GIFs contain multiple images in 256-color GIF format with a specified interval to pause between images. They cannot contain sound. Animated GIFs are supported by many browsers. For more information, see http://www.yahoo.com/Computers_and_Internet/Graphics/Computer_Animation/Animated_GIFs/.

How can I display a watermark?

Both Internet Explorer 2.0 and 3.0 support watermarks. Use the following code:

```
<BODY BACKGROUND="myimage.gif" BGPROPERTIES=FIXED>
```

How can I embed a document or spreadsheet in the middle of my page?

Use a floating frame; for example:

```
<IFRAME WIDTH="75%" HEIGHT="55%" NAME="fframe" SRC="hlsimple.doc">
   <FRAME WIDTH="75%" HEIGHT="55%" NAME="fframe" SRC="hlsimple.doc">
   <EMBED SRC="hlsimple.doc" width=200 height=100>
</IFRAME>
```

To view the embedded document or spreadsheet, the user must have the appropriate application installed. To open hlsimple.doc in the example above, they will need Microsoft Word or Word Viewer, so it may be appropriate to provide a link to http://www.microsoft.com to download a viewer. Viewers are available for Microsoft Word, Microsoft Excel, and Microsoft PowerPoint. You can also convert the document, spreadsheet, or presentation to HTML using appropriate Internet Assistant add-ons. The resulting HTML files may require further enhancements (for example, you may need to use style sheets to retain the original appearance). These HTML files can also be displayed in floating frames as shown above.

Why is a parameter in one of my tags being ignored?

A parameter value that contains non-numeric characters must always be enclosed in quotes. Thus, if you specify size in percentages, you must use quotes (for example, WIDTH="50%"). This rule also applies to any other parameters with non-numeric values, such as HREF. Some browsers will accept some of these values without quotes, but don't count on this always working properly.

How can I start my favorite HTML editor within Internet Explorer?

Although you can't change the program started by the View Source menu item, you can add a customized Edit command button like this:

1. Open Internet Explorer.
2. Choose the View, Options menu item.
3. Select the Programs tab.
4. Click the File Types button.

5. Select "Internet Document (HTML)" in the list box.
6. Click Edit.
7. If an action named Edit is listed, select it and click the Edit button, then proceed to step 10 below.
8. Click New.
9. Type "Edit" in the Action field.
10. Enter the command line needed to execute your desired editor in the "Application used to perform action" field. You need the full path to the executable file, so it's easiest to use Browse to locate it. Most editors require "%1" (including the quotes) after the filename.
11. Click OK or Close as needed to close all open dialog boxes.
12. Exit and restart Internet Explorer.

- ® IE-HTML Authoring Features: http://www.microsoft.com/workshop/author/newfeat/ie30html-f.htm
- ® Frequently Asked Questions About Developing Web Pages Using ActiveX Controls: http://www.microsoft.com/intdev/controls/ctrlfaq-f.htm
- ® A User's Guide to Style Sheets: http://www.microsoft.com/workshop/author/howto/css-f.htm
- ® Author's Guide and HTML Reference: http://www.microsoft.com/workshop/author/newhtml/default.htm
- ® Site Administration: http://www.microsoft.com/workshop/admin/default.htm

Click here to access the most current version of this white paper:
{ewc mvimg, mvimage,!intjump.bmp}

This document provides a description of the mechanism for downloading and installing code for ActiveX objects (components) using the Microsoft ActiveX technologies. This mechanism is used internally by the Microsoft Internet Explorer for downloading ActiveX controls inserted into HTML pages.

Executive Summary

® Internet Component Download is a system-level mechanism for downloading and installing code for ActiveX objects.

® Details are presented for how ActiveX control authors should package their object implementation code so it can be downloaded and installed automatically.

® A new system API, **CoGetClassObjectFromURL**, is presented for downloading ActiveX components. Other "safe code-download" needs can be met using the lower-level **WinVerifyTrust** service, or possibly using a high-level "Setup" ActiveX control. Future releases may expose a more sophisticated component download interface.

® Internet Component Download installs code in a permanent store. This document details a migration strategy for future releases to convert this store into a cache that discards unpopular components.

Note Internet Component Download as specified will *not* download anything other than ActiveX objects. This document does not list steps needed to download/certify other entities. For other code downloading information, see the documentation on the WinVerifyTrust API.

Introduction

Internet Component Download is a system service for downloading, certificate checking, and installing ActiveX component code from the Internet. This service is used by applications (such as Web browsers) to automatically download and install ActiveX objects from code repositories on the Internet. This document explains how code authors should prepare their components for automatic download. It goes on to describe the interface for the component download mechanism and then finally provides some additional implementation details.

Internet Component Download is used within Microsoft Internet Explorer to automatically download ActiveX controls inserted into HTML pages using the `<OBJECT>` element in HTML. (In future releases, code for Document Object components will likewise be downloaded and installed automatically.) The mechanism for downloading components is exposed in an API that may be used in various other ActiveX containers. ActiveX control developers should follow the guidelines outlined below to package their controls so that they can be downloaded automatically by any container that uses the Internet Component Download mechanism.

The following topics are included in this white paper:

Independent software vendors (ISVs) and authors of ActiveX objects for the Internet should package their implementations so that they may be downloaded automatically by Web browsers such as the Microsoft Internet Explorer. Such objects will be downloaded, for instance, when parsing the <OBJECT> tag in HTML. (The <OBJECT> tag used to be called the <INSERT> tag. This change was decided on by the World Wide Web Consortium [W3C] on February 13, 1995.) For details, see the <OBJECT> tag specification. (See the ActiveX Software Development Kit [SDK], which can be installed from the MSDN Development Platform.)

Interpreting the "CODE" URL

The **CODE** attribute in an <OBJECT> tag contains a universal resource locator (URL) pointing to the implementation of a given ActiveX object. This URL is of critical importance for component download, because it must specify all files necessary to implement a particular ActiveX object. HTML authors can author **CODE** URL to point to one of three file types. Component developers should choose one of the packaging schemes below for their ActiveX objects:

1. **A single PE** (portable executable, such as an .OCX or a .DLL): This single executable file is downloaded, installed, and registered in one fell swoop. This is the simplest way to package a single-file ActiveX control, but it will not use file compression, and it will not be platform-independent except with HTTP.
2. **A .CAB** (cabinet) **file**: This file contains one or more files, all of which are downloaded together in a compressed cabinet. (Care must be taken so that the cabinet file contains only those files that must *necessarily* be downloaded [such as the OCX executable itself]. Any additional helper DLLs [such as MFC] may have already been installed and, if so, should not be bundled into the cabinet.) Exactly one file in the cabinet is an .INF file providing further installation information. This .INF file may refer to files in the .CAB as well as to files at other URLs. This mechanism requires authoring of an .INF and packaging of a .CAB file, but in return it provides file compression. It will not be platform-independent, however, except with HTTP format negotiation.
3. **A stand-alone .INF file**: This file specifies various files that need to be downloaded and set up for the OCX to run. The syntax of the .INF file allows URLs pointing to files to download and allows platform independence (by enumerating files for various platforms). This mechanism provides platform independence for non-HTTP servers.

Including version number in the CODE URL

Besides the actual address of code, the **CODE** URL may also include an optional version number using the following syntax:

```
CODE=http://www.foo.com/bar.ocx#Version=a,b,c,d
```

The Internet Component Download mechanism will download and install the file only if the specified version number is more recent than any existing version of the same file currently installed in the system (see the Appendix: Registry Details for more information). If a version number is not specified with a file, it is assumed that any version installed on the system is recent enough.

Platform independence and HTTP

When code to be downloaded is on an HTTP server, the HTTP header MIME request type may be used to specify which platform the code is to run on, thus allowing platform independence of the **CODE** URL.

The following multipurpose Internet mail extensions (MIME) types will be used to describe PEs (portable executables such as an .EXE, .DLL, or .OCX), cabinet files (.CAB), and setup scripts (.INF):

Note The MIME scheme described here is *temporary*. Obviously, this scheme results in too many MIME types. Eventually, MIME attributes will be used for the purpose of describing platform-dependent code (such as **application/x-cabinet; os=win32 cpu=x86**). Until more HTTP servers support such requests, the temporary scheme described above should suffice.

File description	MIME type
PE (portable executable) — .EXE, .DLL, .OCX	**application/x-pe_%opersys%_%cpu%**

Cabinet files—.CAB	**application/x-cabinet_%opersys%_%cpu%**
Setup scripts—.INF (platform independent)	**application/x-setupscript**

%opersys% and **%cpu%** above will specify the operating system and CPU for the desired platform on which the downloaded components will be executed. For example, the MIME type for a Win32 cabinet file running on an Intel *x*86-architecture processor would be **application/x-cabinet_win32_x86**.

The following are valid values for **%opersys%** and **%cpu%**:

Valid values for %opersys%	**Meaning**
win32	32-bit Windows operating systems (Windows 95 or Windows NT)
mac	Macintosh operating system
<other>	Will be defined as necessary
Valid values for %cpu%	Meaning
*x*86	Intel *x*86 family of processors
ppc	Motorola PowerPC architecture
mips	MIPS architecture processors
alpha	DEC Alpha architecture

When the code is on a non-HTTP server (for example, at a local LAN location), an .INF file can be used to achieve platform independence by specifying different URLs for files to be downloaded for different platforms. (See the section below on platform independence in .INF files.)

.CAB format

The .CAB format used for Internet Component Download is a nonproprietary format based on Lempel-Ziv compression. The Microsoft Internet SDK includes a free tool called DIANTZ.EXE that will package cabinet files into this nonproprietary format. There no specification of this .CAB format publicly available, although such a specification will be distributed as soon as possible.

Use of the DIANTZ.EXE tool for creating .CAB cabinet files

The DIANTZ.EXE tool uses a .DDF “directive file” that specifies which files to combine into a cabinet. The syntax for using this tool from a command line is:

```
DIANTZ.EXE /f <directive file.ddf>
```

The example directive file below, CIRC3Z.DDF, would be used for creating a cabinet file containing two files: CIRC3.INF and CIRC3.OCX. It should be a straightforward process to add to this list of files.

```
; DIAMOND directive file for CIRC3.OCX+CIRC3.INF
.OPTION EXPLICIT                    ; Generate errors on variable typos
.Set CabinetNameTemplate=CIRC3Z.CAB
;** The files specified below are stored, compressed, in cabinet files
.Set Cabinet=on
.Set Compress=on
circ3.INF
circ3.OCX
```

.INF setup script format

Here is a sample .INF file that demonstrates the syntax understood by the Internet Component Download

service. Only the .INF syntax below may be used to write setup scripts for Internet Component Download.

```
;Sample INF file for CIRC3.OCX
[Add.Code]
circ3.ocx=circ3.ocx
random.dll=random.dll
mfc40.dll=mfc40.dll
foo.ocx=foo.ocx

[circ3.ocx]
; lines below specify that the specified circ3.ocx (clsid, version) needs to be
installed on
; the system. If doesn't exist already, can be downloaded from the given location
(a .CAB)
file=http://www.code.com/circ3/circ3.cab
clsid={9DBAFCCF-592F-101B-85CE-00608CEC297B}
FileVersion=1,0,0,143

[random.dll]
; lines below specify that the random.dll needs to be installed in the system
; if this doesn't exist already, it can be downloaded from the given location.
file=http:// www.code.com/circ3/random.dll
FileVersion=
DestDir=10
; DestDir can be set to 10 or 11 ( LDID_WIN or LDID_SYS by INF convention)
; this places files in \windows or \windows\system, respectively
; if no dest dir specified (typical case), code is installed in the fixed occache
directory.

[mfc40.dll]
; leaving the file location empty specifies that the installation
; needs mfc40 (version 4,0,0,5), but it should not be downloaded.
; if this file is not already present on the client machine, component download
fails
file=
FileVersion=4,0,0,5

[foo.ocx]
; leaving the file location empty specifies that the installation
; needs the specified foo.ocx (clsid, version), but it should not be downloaded.
; if this file is not already present on the client machine, component download
fails
file=
clsid={DEADBEEF-592F-101B-85CE-00608CEC297B}
FileVersion=1,0,0,143
```

Platform independence in .INF files

It is possible to create platform-independent setup scripts that pull files from different locations depending on the desired platform. Internet Component Download .INF files will use a scheme similar to the one described above under Platform independence and HTTP. Specifically, a sample platform-independent .INF file would include a text such as the following:

```
[circ3.ocx]
; lines below specify that the specified circ3.ocx (clsid, version) needs to be
installed on
; the system. If doesn't exist already, can be downloaded from the given location
(a .CAB)
file_win32_x86=file://products/release/circ3/x86/circ3.cab
```

```
file_win32_mips=file://products/release/circ3/mips/circ3.cab
file_mac_ppc=file://products/release/circ3/macppc/circ3.cab

clsid={9DBAFCCF-592F-101B-85CE-00608CEC297B}
FileVersion=1,0,0,143
```

Thus the **file=** syntax used in the .INF file is expanded to **file_%opersys%_%cpu=**, allowing the .INF file to specify multiple locations where various platform-dependent modules can be found and downloaded. See the section above for valid values for **%opersys%** and **%cpu%**.

The Internet Component Download service is exposed via a single function, **CoGetClassObjectFromURL()**. This system function is called by an application that wishes to download, verify, and install code for an ActiveX component. The function is used in the implementation of Microsoft Internet Explorer. The implementation uses URL monikers to asynchronously download code, and it uses the WinVerifyTrust service to verify validity of the code.

The following are related documents:

® URL Moniker specification: Included in the ActiveX SDK, which can be installed from the MSDN Development Platform.

® Asynchronous Moniker specification: Included in the ActiveX SDK.

® Windows Trust Verification Services specification (describes the **WinVerifyTrust** service mentioned above).

Architecture

The diagram below shows the implementation architecture for the Internet Component Download mechanism and its relation to other system services:
{ewc mvimg, mvimage,!illust.bmp}

Technical Details

This section describes technical details of the Internet Component Download API used by applications (such as Web browsers) to download and install ActiveX object code.

CoGetClassObjectFromURL

This function will return a factory object for a given *rclsid*. If no **CLSID** is specified (**CLSID_NULL**), this function will choose the appropriate **CLSID** for interpreting the Internet MIME type specified in *szContentType*. If the desired object is installed on the system, it is instantiated. Otherwise, the necessary code is downloaded and installed from the location specified in *szCodeURL*.

```
STDAPI CoGetClassObjectFromURL ( [in] REFCLSID rclsid, [in] LPCWSTR szCodeURL,
   [in] DWORD dwFileVersionMS, [in] DWORD dwFileVersionLS,
   [in] LPCWSTR szContentTYPE, [in] LPBINDCTX pBindCtx,
   [in] DWORD dwClsContext, [in] LPVOID pvReserved, [in] REFIID riid,
   [out] VOID **ppv );
```

This "download and install" process involves the following steps:

1. Download the necessary file(s) (.CAB, .INF, or executable) using URL moniker(s).
2. Call **WinVerifyTrust** to ensure that all downloaded files are safe to install.
3. Complete self-registration of all ActiveX components. (Internet Component Download accomplishes self-registration using the **/regserver** command-line argument for .EXE files, and **DLLRegisterServer()** for other executables [such as .DLL and .OCX].)
4. Add registry entries to keep track of downloaded code (see the Appendix: Registry Details).
5. Call **CoGetClassObject** for the desired *rclsid*.

CoGetClassObjectFromURL accepts the following arguments:

Argument	Type	Description
rclsid	**REFCLSID**	**CLSID** of the ActiveX object that needs to be installed. If value is CLSID_NULL, then szContentType is used to determine the **CLSID**.
szCodeURL	**LPCWSTR**	URL pointing to the code for the ActiveX object. This may point to an executable, to an .INF file, or to a .CAB file (see below for details).
dwFileVersionMS	**DWORD**	Major version number for the object

		that needs to be installed.
dwFileVersionLS	**DWORD**	Minor version number for the object that needs to be installed.
szContentType	**LPCWSTR**	MIME type that needs to be understood by the installed ActiveX object. If rclsid is CLSID_NULL, this string is used to determine the **CLSID** of the object that must be installed.
pBindCtx	**LPBINDCTX**	A bind context to use for downloading/installing component code. The client should register its **IBindStatusCallback** in this bind context to receive callbacks during the download and installation process. (See the Asynchronous Monikers specification for details: It's included in the ActiveX SDK, which can be installed from the MSDN Development Platform.)
dwClsContext	**DWORD**	Values taken from the **CLSCTX** enumeration specifying the execution context for the class object.
pvReserved	**LPVOID**	Reserved value, must be set to NULL.
riid	**REFIID**	The interface to obtain on the factory object (typically **IClassFactory**).
ppv	**VOID ****	Pointer in which to store the interface pointer upon return if the call is synchronous.
Returns	**S_OK**	Success. The **ppv** contains the requested interface pointer.
	E_PENDING	Component code will be downloaded and installed asynchronously. The client will receive notifications through the **IBIndStatusCallback** interface it has registered on pBindCtx.
	E_NOINTERFACE	The desired interface pointer is not available. Other **CoGetClassObject** error return values are also possible here.

In the common Web-browser scenario, the values for parameters passed to this function are read directly from an HTML OBJECT tag. For example, the *szCodeURL*, *dwFileVersionMS*, and *dwFileVersionLS* are specified inside an `<OBJECT>` tag as:

```
CODE=http://www.foo.com/bar.ocx#Version=a,b,c,d
```

where: *szCodeURL* is HTTP://WWW.FOO.COM/BAR.OCX, *dwFileVersionMS* is MAKEDWORD(A, B), and *dwFileVersionLS* is MAKEDWORD(C, D).

Because the downloading and installation of code occurs asynchronously, **CoGetClassObjectFromURL** will often return immediately with a return value of E_PENDING. At this point, the **IBindStatusCallBack** mechanism is used to communicate the status of the download operation to the client. (See the Asynchronous Monikers specification for details: It's included in the ActiveX SDK, which can be installed from the MSDN Development Platform.) To participate in this communication, the client *must* implement **IBindStatusCallback** and register this interface in the *pBindCtx* passed into **CoGetClassObjectFromURL** by using **RegisterBindStatusCallback**.

The client can expect to be called with callback notifications for **OnStartBinding** (providing an **IBinding** for controlling the download), **OnProgress** (reporting progress), **OnObjectAvailable** (which returns the desired object interface pointer), and **OnStopBinding** (which returns error codes in case of an error). For further negotiations, the client *must* also implement **ICodeInstall** as described below.

Note The initial (beta) implementation of **CoGetClassObjectFromURL** will not handle system-wide simultaneous downloads of the same code. Similarly, it will not handle cases where different simultaneous downloads refer to the same piece of dependent code.

IBindStatusCallback::OnProgress

The client of **CoGetClassObjectFromURL** will receive notification about the download/install process via the provided **IBindStatusCallback** interface. During the download process, the following additional values (from the BINDSTATUS enumeration) may be passed back as the *ulStatusCode* parameter for **IBindStatusCallback::OnProgress**.

Value	Description
BINDSTATUS_BEGINDOWNLOADCOMPONENTS	The download operation has begun downloading code for ActiveX components that will be installed before the object may be instantiated. The *szStatusText* accompanying **IBindStatusCallback::OnProgress()** provides the display name of the component being downloaded.
BINDSTATUS_INSTALLINGCOMPONENTS	The download operation has downloaded code and is installing it. The *szStatusText* accompanying **IBindStatusCallback::OnProgress()** provides the display name of the component being installed.
BINDSTATUS_ENDDOWNLOADCOMPONENTS	The download operation has finished downloading and installing all necessary code. The *szStatusText* accompanying **OnProgress()** provides the display name of the newly installed component.

ICodeInstall

A code installation operation requires additional services from the client in order to complete the negotiation necessary for a download operation. Such services are requested using **IBindStatusCallback::QueryInterface**. The specific interface requested in **IBindStatusCallback::QueryInterface** is **ICodeInstall**. This interface must be implemented by a client of Internet Component Download.

```
interface ICodeInstall : IUnknown {
   HRESULT   NeedVerificationUI( [out] HWND* phwnd);
   HRESULT   OnCodeInstallProblem( [in] ULONG ulStatusCode,
                                   [in] LPCWSTR szDestination,
                                   [in] LPCWSTR szSource,
```

```
                                    [in] DWORD dwReserved);
};
```

ICodeInstall::NeedVerificationUI

This function is called when Internet Component Download needs to display a user interface (UI) message for verification of downloaded code. (Actually, this UI is displayed by the **WinVerifyTrust** mechanism that is used within Component Download.) When a client is called with this function, it has the opportunity to clear the message queue of its parent window before allowing a UI message to be displayed. If the client does not wish to display the UI message, code verification may continue, but components may fail to be installed.

Argument	Type	Description
phwnd	HWND *	Client-provided HWND of the parent window for displaying code verification UI. If this parameter is NULL, the desktop window is used. If the value is INVALID_HANDLE_VALUE, then no code verification UI will be displayed, and certain necessary components may not be installed.
Returns	S_OK E_INVALIDARG	Success. The argument is invalid.

ICodeInstall::OnCodeInstallProblem

This function is called when there is a problem with code installation. This notification gives the client a chance to resolve the problem, often by displaying a UI message, or by aborting the code installation process. If the client does not understand the problem, it should return E_ABORT by default to abort the code installation process, because returning S_OK would imply retrying the operation.

Argument	Type	Description
ulStatusCode	**ULONG**	Status code describing what problem occurred. A member of CIP_STATUS.
szDestination	**LPCWSTR**	The name of the existing file that was causing a problem. This may be the name of an existing file that needs to be overwritten, the name of a directory causing access problems, or the name of a drive that is full.
szSource	**LPCWSTR**	Name of the new file to replace the existing file (if applicable).
dwReserved	**DWORD**	Reserved for future use.
Returns	S_OK	Continue the installation process. If there was an “access denied” or full-disk problem, retry the installation. If there was an existing file (newer or older version), overwrite it.
	S_FALSE	Skip this particular file,

		but continue with the rest of the code installation process. Note that this is the typical response for the CIP_NEWER_VERSION_EXISTS case.
	E_ABORT	Abort the code installation process.
	E_INVALIDARG	The given arguments are invalid.

The *ulStatusCode* parameter above is one of the following values:

```
typedef enum {
   CIP_DISK_FULL,
   CIP_ACCESS_DENIED,
   CIP_OLDER_VERSION_EXISTS,
   CIP_NEWER_VERSION_EXISTS,
   CIP_NAME_CONFLICT,
   CIP_TRUST_VERIFICATION_COMPONENT_MISSING
} CIP_STATUS;
```

Value	Description
CIP_DRIVE_FULL	The drive specified in *szDestination* is full.
CIP_ACCESS_DENIED	Access to the file specified in *szDestination* is denied.
CIP_OLDER_VERSION_EXISTS	An existing file (older version) specified in *szDestination* needs to be overwritten by the file specified in *szSource*.
CIP_NEWER_VERSION_EXISTS	A file exists (specified in *szDestination*) that is a newer version of a file to be installed (specified in *szSource*).
CIP_NAME_CONFLICT	A file exists (specified in *szDestination*) that has a naming conflict with a file to be installed (specified in *szSource*). The existing file is neither a newer nor an older version of the new file — they are mismatched but have the same filename.
CIP_TRUST_VERIFICATION_COMPONENT_MISSING	The code installation process cannot find the necessary component (**WinVerifyTrust**) for verifying trust in downloaded code. *szSource* specifies the name of the file that cannot be certified. The client should display a UI message asking the user whether or not to install the untrusted code and should then return E_ABORT to abort the download, S_OK to continue anyway, or S_FALSE to skip this file but continue (usually

dangerous).

The Code Download installs most new code in a permanent store in \windows\system\occache. (This directory location is hard-coded for initial releases. In future releases, users may use a registry setting or a Control Panel applet to choose this directory. Component code will be installed in this directory unless a previous version exists. In such cases, the Component Download mechanism will attempt to replace the previous version and invoke **ICodeInstall::OnCodeInstallProblem**.) Some components (helper DLLs that need to be on the system path but currently are not) will also be installed in \windows and \windows\system. All downloaded code is registered using a new Registry "Module Usage" section that keeps track of such code. Downloaded code is *not* removed automatically, but it is possible in the future to add a UI message to the Control Panel (or elsewhere) allowing a user to clean up this directory.

For future releases, it is also possible to convert this "permanent store" into a code cache that retains only popular downloaded code and deletes old unused code automatically. This migration plan justifies use of a permanent store for the first version. See the Appendix: Registry Details for information on how downloaded code is listed in the Registry and how a code cache could function in future releases.

Internet search path

The Internet Component Download service will provide for an Internet “search path” stored in the Registry. This path is a list of Web servers that will be queried every time components are downloaded using **CoGetClassObjectFromURL**. This way, even if an <OBJECT> tag in an HTML document does not specify a location to download code for an embedded ActiveX control, the Internet Component Download will still use the Internet “search path” to find the necessary component.

Further specification of the Internet search path is still under development.

"Pluggable" setup-script handlers

Although Internet Component Download currently supports a limited .INF setup-script syntax, future releases will take into consideration the need to support “hooks” that allow custom setup handlers to interact with the component download and installation process. No further details are available at this time.

There are situations in which code needs to be downloaded with trust verification but the code is not an ActiveX object. Such cases are not addressed by the current specification of the Internet Component Download mechanism. Solutions for these cases need to use the **WinVerifyTrust** mechanism directly, as detailed below:

- **<A HREF> tag in HTML:** It is possible in HTML to download and run .EXE files directly using the <A HREF> tag. The HTML parser uses a URL moniker to download this code, and it calls **WinVerifyTrust** to check validity.
- **Scripts:** Scripting languages will need to define a mechanism for inserting certificates in the script (perhaps in special comments). Given such a mechanism, the **WinVerifyTrust** service will provide trust verification of any such scripts that are downloaded from the Internet.
- **Full applications, other:** The existing Internet Component Download mechanism will not handle extremely complex download situations (such as downloading/installing *DOOM*, registering device drivers, or rebooting the machine). Future releases will aim to allow hooking into the Internet Component Download mechanism to provide more complicated setup routines.

The Internet Component Download service will keep Registry entries for every new downloaded component. These Registry entries will be useful for writing a utility for cleaning up the code storage or migrating the Internet Component Download service to use a code cache rather than a permanent store. (Either of these would be intelligent about uninstalling and deregistering component code using its existing self-registration mechanism.)

Why the existing SharedDLL mechanism is inadequate

To do correct code caching, the existing SharedDLL reference counting scheme will not suffice, because reference counts are easily inflated. Specifically, any application that is reinstalled increases the reference count on a shared DLL even though that DLL already has a reference count belonging to the particular application. (This is already broken for current reference counting, but it will most certainly fail for the Code Download, in which OCXes are used by multiple pages quite regularly, and there is no way of knowing which OCXes need reference counts.)

The new ModuleUsage mechanism in the registry for tracking usage of shared components

To do reference counting correctly, Internet Component Download will maintain a **ModuleUsage** section in the Registry that holds a list of “owners” and “clients” for each shared module. Thus the Registry can keep track of *who* is using a shared module, not just *how many clients* that module has. The Registry entries would use the following syntax:

```
[ModuleUsage]
   [Fully Qualified Path&File Name]
      FileVersion=a,b,c,d
      Owner = Friendly Name/ID of Owner
      [Clients]
         ID of Client1 = <peculiar to this client>
```

A **ModuleUsage** section in a sample Registry would look something like the following:

```
Under My Computer\HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion:

[ModuleUsage]
   [c:\windows/system/mfc40.dll]
      FileVersion=0,4,0,0
      Owner = MSCodeDownloaderID
      [Clients}
         MSCodeDownloaderID= <any info, or default>
         AnotherAppID= <any info, or default>
```

Key name	Description
Fully Qualified Path&Filename	This is the full path of the shared module. This name has to use "/"s instead of "\"s because the "\" is an invalid character in a key name.
Owner	The application that installs the shared module and creates the original **ModuleUsage** section will put an identifier in the owner key section. If the DLL already existed on the system and this Module Usage key did not exist, the owner key should be set to "Unknown" and the DLL should not be removed on uninstallation. The owner should also enlist itself as a client.
File Version	The version number for the shared module.
Clients	IDs of various clients who are using the shared module.

Every client of this module is expected to increment and decrement the existing **SharedDLL** section in the Registry as well (a client only increments this value once when it adds itself as a client under **ModuleUsage**). This allows a migration path for apps currently implementing only a **SharedDLL** scheme.

This Registry information complements the reference counts in the **SharedDLL** section by tracking which clients are actually using a shared module. This counting scheme will work correctly and allow caching of downloaded code. Furthermore, when downloading files, Internet Component Download can use this Registry information as an efficient shortcut for verifying whether a file needs to be overwritten because it is an out-of-date version.

```
<%
If <<MORNING>> Then
    Application("greeting") = "Good Morning"
Else
If <<AFTERNOON>> Then
        Application("greeting") = "Good Afternoon"
    Else
        Application("greeting") = Good Evening"
    End If
End If
%>
```

This section describes various Microsoft programs of interest to Web site developers. The Microsoft Certified Professional program is a series of exams that verify your knowledge of Microsoft products and technologies. By developing software that conforms to Microsoft standards, you can qualify to display Microsoft logos with your products. And, if you want to keep up with the latest Microsoft news and gain valuable business connections, consider joining the Microsoft Solution Provider program.

® Microsoft Certified Professional Program

® Microsoft Logo Program

® Microsoft Solution Provider Program

Click here to connect to the Microsoft Training and Certification Web page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The Microsoft Certified Professional (MCP) program provides the best method to prove your command of current Microsoft products and technologies. Microsoft, an industry leader in certification, is on the forefront of testing methodology. Our exams and corresponding certifications are developed to validate your mastery of critical competencies as you design and develop or implement and support solutions with Microsoft products and technologies. Computer professionals who become Microsoft Certified Professionals are recognized as experts and are sought after industry-wide.

The Microsoft Certified Professional program offers four certifications, based on specific areas of technical expertise:

- ® Microsoft Certified Systems Engineer. MCSEs are qualified to effectively plan, implement, maintain, and support information systems in a wide range of computing environments with Microsoft Windows NT Server and the Microsoft BackOffice integrated family of server products.
- ® Microsoft Certified Solution Developer. MCSDs are qualified to design and develop custom business solutions with Microsoft development tools, technologies, and platforms, including Microsoft Office and Microsoft BackOffice.
- ® Microsoft Certified Product Specialist. MCPSs demonstrate in-depth knowledge of at least one Microsoft operating system. Candidates may pass additional Microsoft certification exams to further qualify their skills with Microsoft BackOffice products, development tools, or desktop applications. See MCPS Areas of Specialization.
- ® Microsoft Certified Trainer. MCTs are instructionally and technically qualified by Microsoft to deliver Microsoft Official Curriculum through Microsoft Solution Provider Authorized Technical Education Centers (ATECs) and Authorized Academic Training Program institutions (AATPs).

Who should become a Microsoft Certified Professional?

Anyone who must demonstrate technical expertise with Microsoft products should consider the program, including systems engineers, professional developers, support technicians, system and network administrators, consultants, and trainers.

What are the benefits of becoming a Microsoft Certified Professional?

The Microsoft Certified Professional program offers a number of immediate and long-term benefits, including the following:

- ® **Recognition.** Microsoft Certified Professionals are instantly recognized as experts with the technical knowledge and skills needed to design and develop or implement and support solutions with Microsoft products. Microsoft helps build this recognition by promoting the expertise of Microsoft Certified Professionals within the industry and to customers and potential clients.
- ® **Access to technical information.** Microsoft Certified Professionals gain access to technical information directly from Microsoft. Depending on the certification, Microsoft Certified Professionals also receive a prepaid trial membership to the Microsoft TechNet Technical Information Network, or a discount for the Microsoft Developer Network; are entitled to free incidents with Microsoft Product Support Services; and are eligible to participate in the Microsoft Beta Evaluation program.
- ® **Global community.** Microsoft Certified Professionals join a worldwide community of technical professionals who have validated their expertise with Microsoft products. Microsoft Certified Professionals are brought together through dedicated forums on MSN, The Microsoft Network; receive a free subscription to *Microsoft Certified Professional Magazine*, a career and professional development magazine created especially for Microsoft Certified Professionals; and, depending on their certification, are eligible to join the Network Professional Association, a worldwide association of computer professionals.

What are the requirements for becoming a Microsoft Certified Professional?

The certification requirements differ for each certification and are specific to the products and job functions addressed by the certification.

To review the requirements, see the description of each certification.

How can I get more information about the Microsoft Certified Professional program?

The Microsoft Training and Certification Web site (see above) offers you and your customers all the information you need regarding Microsoft Official Curriculum courses, Microsoft Certified Professional certifications and exam requirements, and how education supports your customers' certification goals.

Microsoft Train_Cert Offline (formerly Microsoft Roadmap to Education and Certification) is a copy of the Microsoft Training and Certification Web site on CD-ROM. Microsoft Train_Cert Offline is available by calling (800) 636-7544 (United States and Canada).

Exam preparation guides (included in the Microsoft Train_Cert Offline) are available from the Microsoft Sales Fax Service: (800) 727-3351 (refer to the document catalog).

Sample assessment exams (included in the Microsoft Train_Cert Offline) are available by calling (800) 636-7544 (United States and Canada).

The Exam Study Guide provides information about how Microsoft Certified Professional exams are developed. It is available from the Microsoft Sales Fax Service: (800) 727-3351 (refer to the document calalog).

Click here to connect to the Microsoft Training and Certification page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The Microsoft Certified Professional (MCP) program provides the best method to prove your command of current Microsoft products and technologies. Microsoft, an industry leader in certification, is on the forefront of testing methodology. Our exams and corresponding certifications are developed to validate your mastery of critical competencies as you design and develop or implement and support solutions with Microsoft products and technologies. Computer professionals who become Microsoft Certified Professionals are recognized as experts and are sought after industrywide.

For network professionals, Microsoft offers the Microsoft Certified Systems Engineer (MCSE) credential. Microsoft Certified Systems Engineers are qualified to effectively plan, implement, maintain, and support information systems in a wide range of computing environments with Microsoft Windows NT Server and the Microsoft BackOffice integrated family of server products.

To become a Microsoft Certified Systems Engineer, you must pass a series of rigorous certification exams that provide a valid and reliable measure of technical proficiency and expertise. These exams are developed with the input of professionals in the industry and reflect how Microsoft products are used in organizations throughout the world. The exams are administered by an independent organization — Sylvan Prometric — at more than 800 Authorized Prometric Testing Centers around the world.

Benefits of Becoming a Microsoft Certified Systems Engineer

Why should you become a Microsoft Certified Systems Engineer? To demonstrate to your customers and colleagues that you have what it takes to plan, implement, and support business solutions with Microsoft Windows NT and Microsoft BackOffice. As a Microsoft Certified Systems Engineer, you receive the following benefits:

® Industry recognition of your knowledge and proficiency with Microsoft products and technologies.

® Access to technical information directly from Microsoft. Microsoft Certified Systems Engineers receive a complimentary one-year subscription to the Microsoft TechNet Technical Information Network, providing valuable information via monthly CDs. If you currently have a Microsoft TechNet subscription, an additional year will be added to your existing subscription, free of charge.

® One free Priority Comprehensive support 10 pack from Microsoft and a 25 percent discount on purchases of additional Priority Comprehensive support incident 10 packs. You gain access to support for any Microsoft product, 24 hours a day, seven days a week. This provides the opportunity — when you need it — to resolve customer issues with the help of Microsoft support engineers.

® A one-year subscription to the Microsoft Beta Evaluation program. This benefit provides you with up to 12 free monthly CDs containing beta software (English only) for many of Microsoft's newest software products, allowing you to become familiar with new versions of Microsoft products before they are generally available.

® A free annual subscription to *Microsoft Certified Professional Magazine*, a career and professional development magazine created especially for Microsoft Certified Professionals.

® Microsoft Certified Systems Engineer logos and other materials to let you identify your Microsoft Certified Systems Engineer status to colleagues or clients.

® A Dedicated forum on MSN, The Microsoft Network, that allows Microsoft Certified Professionals to communicate directly with Microsoft and one another.

® *Certification Update* bimonthly newsletter from the Microsoft Certified Professional program, keeping you informed of changes and advances in the program and exams.

® Invitations to Microsoft conferences, technical training sessions, and special events.

Microsoft Certified Systems Engineer Requirements

Microsoft Certified Systems Engineers are required to pass four operating system exams and two elective exams. The operating system exams require candidates to prove their expertise with desktop, server, and networking components. The elective exams require proof of expertise with Microsoft BackOffice products.

For specific MCSE requirements, please visit the Certified Systems Engineer Web page (see below) or see the

Microsoft Train_Cert Offline (formerly Microsoft Roadmap to Education and Certification), a copy of the Microsoft Training and Certification Web site on CD-ROM. Microsoft Train_Cert Offline is available by calling (800) 636-7544 (United States and Canada).

Click here to connect to the Microsoft Certified Systems Engineer page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Process for Becoming a Microsoft Certified Systems Engineer

® **Preparation.** Exam preparation guides with outlines of exam topics and lists of relevant training resources are available on the Microsoft Training and Certification Web site (see top of page) and included in Microsoft Train_Cert Offline. Sample assessment exams, which enable individuals to evaluate skills and practice answering questions similar to those on actual exams, are also available.

® **Training.** Microsoft Official Curriculum courses can help you gain the skills needed for certification. These courses are available through Microsoft Solution Provider Authorized Technical Education Centers (ATECs) in both instructor-led and self-paced formats. For referrals to Microsoft ATECs, call (800) SOL-PROV.

® **Exams.** Sylvan Prometric administers all Microsoft Certified Professional exams. You must register with Sylvan Prometric prior to scheduling an exam. Exams may be taken at any of more than 800 Authorized Prometric Testing Centers around the world. To register for an exam, please call Sylvan Prometric at (800) 755-EXAM.

For More Information

For additional information on becoming a Microsoft Certified Systems Engineer, or on Microsoft Training and Certification in general, visit the Microsoft Training and Certification Web site (see top of page) or get a copy of Microsoft Train_Cert Offline. Both resources include complete information about the Microsoft Certified Professional program and Microsoft Official Curriculum and sample exams for the MCP program.

Microsoft Train_Cert Offline is available by calling (800) 636-7544 (United States and Canada).

Exam preparation guides (included in the Microsoft Train_Cert Offline) are available from the Microsoft Sales Fax Service: (800) 727-3351 (refer to the document catalog).

Sample assessment exams (included in the Microsoft Train_Cert Offline) are available by calling (800) 636-7544 (United States and Canada).

The Exam Study Guide, which provides information about how Microsoft Certified Professional exams are developed, is available by using the Microsoft Sales Fax Service: (800) 727-3351 (request document catalog #1000-0733).

Click here to connect to the Microsoft Training and Certification page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The Microsoft Certified Professional (MCP) program provides the best method to prove your command of current Microsoft products and technologies. Microsoft, an industry leader in certification, is on the forefront of testing methodology. Our exams and corresponding certifications are developed to validate your mastery of critical competencies as you design and develop or implement and support solutions with Microsoft products and technologies. Computer professionals who become Microsoft Certified Professionals are recognized as experts and are sought after industrywide.

For developers, Microsoft offers the Microsoft Certified Solution Developer (MCSD) credential. Microsoft Certified Solution Developers are qualified to design and develop custom business solutions with Microsoft development tools, technologies, and platforms, including Microsoft Office and Microsoft BackOffice.

To become a Microsoft Certified Solution Developer, you must pass a series of rigorous certification exams that provide a valid and reliable measure of technical proficiency and expertise. These exams are developed with the input of professionals in the industry and reflect how Microsoft products are used in organizations throughout the world. The exams are administered by an independent organization — Sylvan Prometric — at more than 800 Authorized Prometric Testing Centers around the world.

Benefits of Becoming a Microsoft Certified Solution Developer

Why should you become a Microsoft Certified Solution Developer? To demonstrate to your customers and colleagues that you have what it takes to design and develop superior custom solutions with Microsoft tools and technologies. As a Microsoft Certified Solution Developer, you receive the following benefits:

® Industry recognition of your knowledge and proficiency with Microsoft products and technologies.

® Microsoft Developer Network subscription discount. Microsoft Certified Solution Developers receive a discount they may apply toward a one-year Microsoft Developer Network (MSDN) Library Subscription or Professional Subscription, providing valuable information on quarterly CDs.

® One free 10 pack of Priority Development with Desktop support incidents and a 25 percent discount on purchases of additional Priority Developer and Desktop support incident 10 packs. You gain access to support for any Microsoft product, seven days a week, 24 hours a day. This provides the opportunity-when you need it-to resolve customer issues with the help of Microsoft support engineers.

® A one-year subscription to the Microsoft Beta Evaluation program. This benefit provides you with up to 12 free monthly CDs containing beta software (English only) for many of Microsoft's newest software products. This allows you to become familiar with new versions of Microsoft products before they are generally available.

® A free annual subscription to *Microsoft Certified Professional Magazine*, a career and professional development magazine created especially for Microsoft Certified Professionals.

® Microsoft Certified Solution Developer logos and other materials to help you identify your Microsoft Certified Solution Developer status to colleagues or clients.

® A dedicated forum on MSN, The Microsoft Network, that enables Microsoft Certified Professionals to communicate directly with Microsoft and one another.

® *Certification Update* bimonthly newsletter from the Microsoft Certified Professional program, keeping you informed of changes and advances in the program and exams.

® Invitations to Microsoft conferences, technical training sessions, and special events.

Microsoft Certified Solution Developer Requirements

Microsoft Certified Solution Developers are required to pass two core-technology exams and two elective exams. The core technology exams require candidates to prove their understanding of Windows 32-bit architecture, OLE, UI design, and Windows Open Services Architecture components. The elective exams require proof of expertise with Microsoft development tools.

For specific MCSD requirements, please visit the Certification Solution Developers Web page (see below) or see Microsoft Train_Cert Offline (formerly Microsoft Roadmap to Education and Certification), a copy of the Microsoft Training and Certification Web site on CD-ROM. Microsoft Train_Cert Offline is available by calling (800) 636-

7544 (United States and Canada).

Click here to connect to the Certified Solution Developers Web page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Process for Becoming a Microsoft Certified Solution Developer

® **Preparation.** Exam preparation guides with outlines of exam topics and lists of relevant training resources are available on the Microsoft Training and Certification Web site (see top of page) and included in Microsoft Train_Cert Offline. Sample assessment exams, which enable individuals to evaluate skills and practice answering questions similar to those on actual exams, are also available.

® **Training.** Microsoft Official Curriculum courses can help you gain the skills needed for certification. These courses are available through Microsoft Solution Provider Authorized Technical Education Centers (ATECs) in both instructor-led and self-paced formats. For referrals to Microsoft ATECs, call (800) SOL-PROV.

® **Exams.** Sylvan Prometric administers all Microsoft Certified Professional exams. You must register with Sylvan Prometric prior to scheduling an exam. Exams may be taken at any of more than 800 Authorized Prometric Testing Centers around the world. To register for an exam, please call Sylvan Prometric at (800) 755-EXAM.

For More Information

For additional information on becoming a Microsoft Certified Solution Developer, or on Microsoft Training and Certification in general, visit the Microsoft Training and Certification Web site (see top of page) or get a copy of Microsoft Train_Cert Offline. Both resources include complete information about the Microsoft Certified Professional program and Microsoft Official Curriculum and sample exams for the MCP program.

Microsoft Train_Cert Offline is available by calling (800) 636-7544 (United States and Canada).

Exam preparation guides are available by using the Microsoft Sales Fax Service: (800) 727-3351 (refer to the document catalog).

Sample assessment exams are available by calling (800) 636-7544 (United States and Canada).

The Exam Study Guide, which provides information about how Microsoft Certified Professional exams are developed, is available by using the Microsoft Sales Fax Service: (800) 727-3351 (request document catalog #1000-0733).

Click here to connect to the Microsoft Training and Certification Web page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The Microsoft Certified Professional (MCP) program provides the best method to prove your command of current Microsoft products and technologies. Microsoft, an industry leader in certification, is on the forefront of testing methodology. Our exams and corresponding certifications are developed to validate your mastery of critical competencies as you design and develop or implement and support solutions with Microsoft products and technologies. Computer professionals who become Microsoft Certified Professionals are recognized as experts and are sought after industrywide.

For individuals who would like to demonstrate their expertise with a particular Microsoft product, Microsoft offers the Microsoft Certified Product Specialist (MCPS) credential. Microsoft Certified Product Specialists have demonstrated in-depth knowledge of at least one Microsoft operating system. Candidates may pass additional Microsoft certification exams to further qualify their skills with Microsoft BackOffice products, development tools, or desktop applications.

To become a Microsoft Certified Product Specialist, you must pass one or more rigorous certification exams that provide a valid and reliable measure of technical proficiency and expertise. These exams are developed with the input of professionals in the industry and reflect how Microsoft products are used in organizations throughout the world. The exams are administered by an independent organization — Sylvan Prometric — at more than 800 Authorized Prometric Testing Centers around the world.

Benefits of Becoming a Microsoft Certified Product Specialist

Why should you become a Microsoft Certified Product Specialist? To demonstrate to your customers and colleagues that you have the specialized knowledge required to use or support specific Microsoft products. As a Microsoft Certified Product Specialist, you receive the following benefits:

® Industry recognition of your knowledge and proficiency with Microsoft products and technologies.

® Access to technical information directly from Microsoft. Microsoft Certified Product Specialists receive a complimentary Microsoft TechNet CD, plus a 50 percent discount toward a one-year membership to the Microsoft TechNet Technical Information network, providing valuable information on monthly CDs.

® A free annual subscription to *Microsoft Certified Professional Magazine*, a career and professional development magazine created especially for Microsoft Certified Professionals.

® Microsoft Certified Product Specialist logos and other materials to enable you to identify your Microsoft Certified Product Specialist status to colleagues or clients.

® A dedicated forum on MSN, The Microsoft Network, that enables Microsoft Certified Professionals to communicate directly with Microsoft and one another.

® *Certification Update* bimonthly newsletter from the Microsoft Certified Professional program, keeping you informed of changes and advances in the program and exams.

® Invitations to Microsoft conferences, technical training sessions, and special events.

Microsoft Certified Product Specialist Requirements

Microsoft Certified Product Specialists are required to pass one operating system exam, proving their expertise with a current Microsoft Windows desktop or server operating system.

For specific MCPS requirements, please visit the Certified Product Specialist Web page (see below) or see the Microsoft Train_Cert Offline (formerly Microsoft Roadmap to Education and Certification), a copy of the Microsoft Training and Certification Web site on CD-ROM. Microsoft Train_Cert Offline is available by calling (800) 636-7544 (United States and Canada).

Click here to connect to the Certified Product Specialist Web page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Process for Becoming a Microsoft Certified Product Specialist

- **Preparation.** Exam preparation guides with outlines of exam topics and lists of relevant training resources are available at the Microsoft Training and Certification Web site (see top of page). Sample assessment exams, which enable individuals to evaluate skills and practice answering questions similar to those on actual exams, are also available. In addition, an Exam Study Guide is available that provides information about how Microsoft Certified Professional exams are developed. Microsoft Train_Cert Offline (formerly Microsoft Roadmap to Education and Certification) is a copy of the Microsoft Training and Certification Web site on CD-ROM. Microsoft Train_Cert Offline is available by calling (800) 636-7544 (United States and Canada).
- **Training.** Microsoft Official Curriculum courses can help you gain the skills needed for certification. These courses are available through Microsoft Solution Provider Authorized Technical Education Centers (ATECs) in both instructor-led and self-paced formats. For referrals to Microsoft ATECs, call (800) SOL-PROV.
- **Exams.** Sylvan Prometric administers all Microsoft Certified Professional exams. You must register with Sylvan Prometric prior to scheduling an exam. Exams may be taken at any of more than 800 Authorized Prometric Testing Centers around the world. To register for an exam, please call Sylvan Prometric at (800) 755-EXAM.

For More Information

The Microsoft Training and Certification Web site (see top of page) offers you and your customers all the information you need regarding Microsoft Official Curriculum courses, Microsoft Certified Professional certifications and exam requirements, and how education supports your customers' certification goals.

Microsoft Train_Cert Offline (formerly Microsoft Roadmap to Education and Certification) is a copy of the Microsoft Training and Certification Web site on CD-ROM. Microsoft Train_Cert Offline is available by calling (800) 636-7544 (United States and Canada).

Exam preparation guides (included in the Microsoft Train_Cert Offline) are available from the Microsoft Sales Fax Service: (800) 727-3351 (refer to the document catalog).

Sample assessment exams (included in the Microsoft Train_Cert Offline) are available by calling (800) 636-7544 (United States and Canada).

The Exam Study Guide provides information about how Microsoft Certified Professional exams are developed. It is available from the Microsoft Sales Fax Service: (800) 727-3351 (refer to the document catalog).

An area of specialization should focus on a specific Microsoft product or technology. For an area of specialization, the exam combination should include a minimum of one required operating system exam and one elective exam. The attached document, Microsoft Certified Product Specialist Examples of Areas of Specialization, contains some examples of areas of specialization identified by Microsoft that can be used to guide you further.

Below is a list of areas of specialization identified by Microsoft that can be used to guide you in combining exams. These specializations are categorized into three tracks:

® **Systems Engineer** combinations are chosen from Microsoft Certified Systems Engineer requirements.

® **Solution Developer** combinations are chosen from Microsoft Certified Solution Developer requirements.

® **Desktop Support** combinations include a desktop operating system exam and any desktop application exam.

Systems Engineer

Area of Specialization: Networking — Leads to Systems Engineer

Operating System Exam

® Exam 70-67: Implementing and Supporting Microsoft Windows NT Server 4.0

OR

® Exam 70-43: Implementing and Supporting Microsoft Windows NT Server 3.51

Elective Exams (two exams: one desktop operating system exam and one networking exam)

Choose one desktop operating system exam:

® Exam 70-63: Implementing and Supporting Microsoft Windows 95

® Exam 70-30: Microsoft Windows 3.1

® Exam 70-48: Microsoft Windows for Workgroups 3.11 — Desktop

® Exam 70-73: Implementing and Supporting Microsoft Windows NT Workstation 4.0

OR

® Exam 70-42: Implementing and Supporting Microsoft Windows NT Workstation 3.51

Choose one networking exam:

® Exam 70-58: Networking Essentials

® Exam 70-47: Networking with Microsoft Windows 3.1

® Exam 70-46: Networking with Microsoft Windows for Workgroups 3.11 — Desktop

Area of Specialization: Microsoft TCP/IP — Leads to Systems Engineer

Operating System Exam

® Exam 70-67: Implementing and Supporting Microsoft Windows NT Server 4.0

OR

® Exam 70-43: Implementing and Supporting Microsoft Windows NT Server 3.51

Elective Exam

® Exam 70-53: Internetworking Microsoft TCP/IP on Microsoft Windows NT (3.5–3.51)

Area of Specialization: Microsoft Mail for PC Networks 3.2 — Leads to Systems Engineer

Operating System Exam

® Exam 70-67: Implementing and Supporting Microsoft Windows NT Server 4.0

OR

® Exam 70-43: Implementing and Supporting Microsoft Windows NT Server 3.51

Elective Exam

® Exam 70-37: Microsoft Mail for PC Networks 3.2 — Enterprise

Area of Specialization: Microsoft SQL Server 4.2 — Leads to Systems Engineer

Operating System Exam

® Exam 70-67: Implementing and Supporting Microsoft Windows NT Server 4.0

OR

® Exam 70-43: Implementing and Supporting Microsoft Windows NT Server 3.51

Elective Exams (two exams)

® Exam 70-26: System Administration of Microsoft SQL Server 6

® Exam 70-27: Implementing a Database Design on Microsoft SQL Server 6

OR

® Exam 70-21: Microsoft SQL Server 4.2 Database Implementation

® Exam 70-22: Microsoft SQL Server 4.2 Database Administration for Microsoft Windows NT

Area of Specialization: Microsoft Systems Management Server 1.0 — Leads to Systems Engineer

Operating System Exam

® Exam 70-67: Implementing and Supporting Microsoft Windows NT Server 4.0

OR

® Exam 70-43: Implementing and Supporting Microsoft Windows NT Server 3.51

Elective Exam

® Exam 70-14: Implementing and Supporting Microsoft Systems Management Server 1.0

Area of Specialization: Microsoft SNA Server — Leads to Systems Engineer

Operating System Exam

® Exam 70-67: Implementing and Supporting Microsoft Windows NT Server 4.0

OR

® Exam 70-43: Implementing and Supporting Microsoft Windows NT Server 3.51

Elective Exam

® Exam 70-12: Microsoft SNA Server

Area of Specialization: Microsoft Exchange — Leads to Systems Engineer

Operating System Exam

® Exam 70-67: Implementing and Supporting Microsoft Windows NT Server 4.0

OR

® Exam 70-43: Implementing and Supporting Microsoft Windows NT Server 3.51

Elective Exam

® Exam 70-75: Implementing and Supporting Microsoft Exchange

Solution Developer

Area of Specialization: Microsoft Visual Basic 3.0 for Windows — Leads to Solution Developer

Operating System Exam

Choose one Microsoft Windows architecture exam:

® Exam 70-150: Microsoft Windows Operating Systems and Services Architecture I

® Exam 70-151: Microsoft Windows Operating Systems and Services Architecture II

Elective Exam

® Exam 70-50: Microsoft Visual Basic 3.0 for Windows — Application Development

Area of Specialization: Microsoft Access 2.0 for Windows — Leads to Solution Developer

Operating System Exam

Choose one Microsoft Windows architecture exam:

® Exam 70-150: Microsoft Windows Operating Systems and Services Architecture I

® Exam 70-151: Microsoft Windows Operating Systems and Services Architecture II

Elective Exam

® Exam 70-51: Microsoft Access 2.0 for Windows — Application Development

Area of Specialization: Microsoft Excel Application Development — Leads to Solution Developer

Operating System Exam

Choose one Microsoft Windows architecture exam:

® Exam 70-150: Microsoft Windows Operating Systems and Services Architecture I

® Exam 70-151: Microsoft Windows Operating Systems and Services Architecture II

Elective Exam

® Exam 70-52: Developing Applications with Microsoft Excel 5.0 Using Visual Basic for Applications

Area of Specialization: Microsoft Visual FoxPro — Leads to Solution Developer

Operating System Exam

Choose one Microsoft Windows architecture exam:

® Exam 70-150: Microsoft Windows Operating Systems and Services Architecture I

® Exam 70-151: Microsoft Windows Operating Systems and Services Architecture II

Elective Exam

® Exam 70-54: Programming in Microsoft Visual FoxPro 3.0 for Windows

Area of Specialization: Implementing OLE in MFC Applications — Leads to Solution Developer

Operating System Exam

Choose one Microsoft Windows architecture exam:

® Exam 70-150: Microsoft Windows Operating Systems and Services Architecture I

® Exam 70-151: Microsoft Windows Operating Systems and Services Architecture II

Elective Exam

® Exam 70-25: Implementing OLE in Microsoft Foundation Class Applications

Area of Specialization: Microsoft Visual Basic 4.0 — Leads to Solution Developer

Operating System Exam

Choose one Microsoft Windows architecture exam:

® Exam 70-150: Microsoft Windows Operating Systems and Services Architecture I

® Exam 70-151: Microsoft Windows Operating Systems and Services Architecture II

Elective Exam

® Exam 70-65: Programming with Microsoft Visual Basic 4.0

Area of Specialization: Microsoft Access for Windows 95 — Leads to Solution Developer

Operating System Exam

Choose one Microsoft Windows architecture exam:

® Exam 70-150: Microsoft Windows Operating Systems and Services Architecture I

® Exam 70-151: Microsoft Windows Operating Systems and Services Architecture II

Elective Exam

® Exam 70-69: Microsoft Access for Windows 95 and the Microsoft Access Developer's Toolkit

Desktop Support

Area of Specialization: Microsoft Word 6.0 for Windows — Leads to Desktop Support

Operating System Exam

Choose one desktop operating system exam:

® Exam 70-63: Microsoft Windows 95

® Exam 70-30: Microsoft Windows 3.1

® Exam 70-48: Microsoft Windows for Workgroups 3.11 — Desktop

® Exam 70-73: Implementing and Supporting Microsoft Windows NT Workstation 4.0

OR

® Exam 70-42: Implementing and Supporting Microsoft Windows NT Workstation 3.51

Elective Exam

® Exam 70-49: Microsoft Word 6.0 for Windows

OR

® Exam 70-66: Microsoft Word for Windows 95

Area of Specialization: Microsoft Excel 5.0 for Windows — Leads to

Desktop Support

Operating System Exam

Choose one desktop operating system exam:

® Exam 70-63: Microsoft Windows 95

® Exam 70-30: Microsoft Windows 3.1

® Exam 70-48: Microsoft Windows for Workgroups 3.11 — Desktop

® Exam 70-73: Implementing and Supporting Microsoft Windows NT Workstation 4.0

OR

® Exam 70-42: Implementing and Supporting Microsoft Windows NT Workstation 3.51

Elective Exam

® Exam 70-39: Microsoft Excel 5.0 for Windows

Area of Specialization: Microsoft Project 4.0 for Windows — Leads to Desktop Support

Operating System Exam

Choose one desktop operating system exam:

® Exam 70-63: Implementing and Supporting Microsoft Windows 95

® Exam 70-30: Microsoft Windows 3.1

® Exam 70-48: Microsoft Windows for Workgroups 3.11 — Desktop

® Exam 70-73: Implementing and Supporting Microsoft Windows NT Workstation 4.0

OR

® Exam 70-42: Implementing and Supporting Microsoft Windows NT Workstation 3.51

Elective Exam

® Exam 70-38: Microsoft Project 4.0 for Windows

Logo recognition for Areas of Specialization

Candidates who pass a minimum of one required operating system exam and one elective are encouraged to display their specialty as an additional tagline below their Microsoft Certified Product Specialist logo. Examples of the MCPS logo including Area of Specialization taglines include:

Area of Specialization:	**Area of Specialization:**
Networking	Microsoft Visual Basic 3.0
{ewc MVIMG, MVIMAGE,!mscert.bmp}	{ewc MVIMG, MVIMAGE,!mscert.bmp}
Product Specialist Networking	*Product Specialist Microsoft Visual Basic 3.0*
Area of Specialization:	**Area of Specialization:**
Microsoft Word 6.0 for Windows	Multiple Specializations
{ewc MVIMG, MVIMAGE,!mscert.bmp}	{ewc MVIMG, MVIMAGE,!mscert.bmp}
Product Specialist Microsoft Word 6.0 for Windows	*Product Specialist Networking Microsoft Visual Basic 3.0 Microsoft Word 6.0 for Windows*

Click here to connect to the Microsoft Training and Certification Web page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Microsoft Certified Trainers (MCTs) play an important role in Microsoft's Education and Certification process. MCTs are qualified instructionally and certified technically by Microsoft to deliver Microsoft Official Curriculum instructor-led courses to computer professionals. Only trainers who will to deliver Microsoft Official Curriculum instructor-led courses at Microsoft Authorized Technical Education Centers are eligible to become Microsoft Certified Trainers. Once you have been approved as an MCT, your MCT credential will be accepted in all Microsoft education channels.

Microsoft Official Curriculum courses, developed by Microsoft product groups, educate computer professionals who develop, support, and implement solutions that use Microsoft technology.

Microsoft authorized education channels include Microsoft Authorized Technical Education Centers (ATECs), Microsoft Authorized Academic Training Program (AATP), and Microsoft Online Institute (MOLI).

For more information about the process of becoming an MCT, refer to the Microsoft Certified Trainer Guide available for downloading at the Microsoft Training and Certification Web site (see top of page) or by calling (800) 636-7544. After you have reviewed this document, complete a Microsoft Certified Trainer Application. To request an MCT application kit, please call Microsoft at (800) 688-0496.

Benefits of Becoming a Microsoft Certified Trainer

Why should you become a Microsoft Certified Trainer? To qualify to deliver Microsoft Official Curriculum at Microsoft authorized education sites. Only trainers who intend to deliver Microsoft Official Curriculum at Microsoft authorized education sites are eligible to become Microsoft Certified Trainers. As a Microsoft Certified Trainer, you receive the following benefits:

® Industry recognition of your knowledge and proficiency with Microsoft products and technologies, and expertise delivering Microsoft Official Curriculum.

® Access to technical information directly from Microsoft. MCTs receive a complimentary Microsoft TechNet CD, plus a 50 percent discount toward a one-year membership to the *Microsoft TechNet* Technical Information Network, providing valuable information on monthly CDs.

® Microsoft Certified Trainer logos and other materials to let you identify your Microsoft Certified Trainer status to colleagues or clients.

® A dedicated MCT Forum on MSN, The Microsoft Network, that enables Microsoft Certified Trainers to communicate directly with Microsoft trainers and course developers, and one another.

® Annual subscription to *Microsoft Certified Professional Magazine,* a career and professional development magazine created especially for Microsoft Certified Professionals.

® *Education Forum* monthly newsletter from Microsoft Education and Certification, keeping you informed of new courses, new exams, trainer certification requirements, course schedule, and program updates.

® *Certification Update* bimonthly newsletter from the Microsoft Certified Professional program, keeping you informed of changes and advances in the program and exams.

® Invitations to Microsoft conferences, technical training sessions, and special events.

Microsoft Certified Trainer Requirements

Microsoft Certified Trainers are certified on a course-by-course basis. For specific trainer certification requirements for each Microsoft Official Curriculum refer to the *Microsoft Certified Trainer Course Requirements* in the *Microsoft Certified Trainer Information Kit* or use the Microsoft Sales Fax Service at (800) 727-3351.

Process for Becoming a Microsoft Certified Trainer

® **Apply.** Submit a Microsoft Certified Trainer application with proof of your instructional skills and the name of the Microsoft authorized education site that will employ you (or to which you are under contract).

® **Attend the Course.** Attend the Microsoft Official Curriculum course at a local Microsoft ATEC or attend a Microsoft Certified Trainer course at Microsoft. For referrals to Microsoft ATECs, call (800) SOLPROV. Fax your student course certificate to Microsoft at (801) 578-1429 to demonstrate you have attended the course.

- ® **Prepare for the Exam.** Exam preparation guides with outlines of exam topics and lists of relevant training resources are available on the Microsoft Training and Certification Web site (see top of page) and included in Microsoft Train_Cert Offline. Sample assessment exams, which enable individuals to evaluate skills and practice answering questions similar to those on actual exams, are also available.
- ® **Pass the Exam(s).** Sylvan Prometric administers all Microsoft Certified Professional exams and Microsoft trainer exams. When there is no Microsoft Certified Professional exam for a Microsoft Official Curriculum course, a trainer exam is created. You must register with Sylvan prior to scheduling an exam. You may take exams at any of more than 700 Sylvan Authorized Testing Centers around the world. To register for an exam, call Sylvan at (800) 755-EXAM.
- ® **Purchase and Review the Trainer Kit.** Prepare to deliver the Microsoft Official Curriculum course by calling Microsoft at (800) 688-0496 to purchase the Microsoft Trainer Kit. Outside the United States and Canada, trainers can purchase trainer kits through their local Microsoft ATEC. Review all Microsoft Trainer Kit materials and course materials, and complete all labs and demonstrations.

For More Information

For additional information on becoming a Microsoft Certified Trainer or to receive a *Microsoft Certified Trainer Information Kit,* call Microsoft at (800) 688-0496. For information regarding Microsoft Training and Certification in general, visit the Web site (see top of page).

Microsoft Train_Cert Offline (formerly Microsoft Roadmap to Education and Certification) is a copy of the Microsoft Training and Certification Web site on CD-ROM. Microsoft Train_Cert Offline is available by calling (800) 636-7544 (United States and Canada).

Exam preparation guides are available separately by using the Microsoft Sales Fax Service: (800) 727-3351 (refer to the document catalog).

Sample assessment exams are available by calling (800) 636-7544 (United States and Canada).

An Exam Study Guide, which provides information about how Microsoft Certified Professional exams are developed, is available by using the Microsoft Sales Fax Service: (800) 727-3351 (request document catalog #1000-0733).

This section contains information about the Microsoft Logo Programs. The Microsoft logo assures end users that your product takes advantage of the new technologies integrated into all Microsoft products.

® Powered by Microsoft BackOffice Logo Program

® Windows 95 Logo Program

For up-to-date information, see the Powered by BackOffice page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Until recently, Web pages were mainly static HTML pages, providing a way for businesses to simply present information on products or services. Today, businesses recognize that web sites can do much more. Innovative Web sites allow people to dynamically interact with information, find what they want, and enjoy the experience. In short, the best Web sites are destinations worth visiting over and over.

In its commitment to enable the evolution of the Internet, Microsoft delivers powerful solutions for businesses to customize, manage and present dynamic information on the Web while meeting customers' demands for speed and stability. Through the power of the Microsoft BackOffice suite of integrated server products including the Internet Information Server, businesses now have the best platform for the new generation of dynamic web applications.

Web sites bearing the Powered by Microsoft BackOffice logo utilize the versatility of these powerful solutions, setting a new standard for excellence in web development. Below are some examples of sites Powered by BackOffice on the Web today.

Database-driven publishing

The Internet Database Connector in Microsoft's Internet Information Server provides seamless access to ODBC databases, such as Microsoft SQL Server and Microsoft Access for the fastest way to publish database information on the Internet. Microsoft's Online Events Calendar demonstrates how easy this can be when your site is Powered by BackOffice.

Web sites bearing the Powered by Microsoft BackOffice logo utilize the versatility of these powerful solutions, setting a new standard for excellence in Web development.

Click here to connect to the Designed for Windows 95 Logo page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The Designed for Windows 95 logo was developed to help end users easily identify hardware and software products that were designed specifically for the Microsoft Windows 95 operating system. Users can mix and match hardware and software products designated with the Windows 95 logo and be assured that the products will take advantage of the new technologies integrated into the Microsoft Windows 95 operating system. Licensing the logo enables you to use the logo on your product packaging, advertising, collateral, and other marketing materials, which signals to customers that you designed your product to give them extra benefits when it is run on Windows 95.

Because both hardware and software are now tested before rights to use the logo are granted, customers have a high regard for the logo. Hardware is tested by the Microsoft Windows Hardware Quality Labs (whqlinfo@microsoft.com). Software applications are tested by a fully independent laboratory, VeriTest, Inc. (logolab@VeriTest.com).

Important Notes

The logo indicates that a software product provides all the features outlined in these guidelines; it is not a "quality assurance" seal. VeriTest will not be testing the quality of your product or ensuring that it is "bug-free" — VeriTest's job is just to make sure that your product has full functionality, that the logo features are present, and that your product is not generating frequent faults or system crashes.

Please note that the license agreement states:

You may only use the Logo as a symbol that your Product is compatible with the Microsoft Windows 95 operating system. You may not explicitly state or imply that Microsoft in any way endorses your product. Also, the Windows Logo program is not intended to be a "certification" program, i.e., the Logo does not represent that Microsoft certifies your product in any way.

Please note further that VeriTest's role in the process is to test applications for compliance with the logo criteria. A passing test result is not a certification or endorsement of your product(s) by VeriTest.

Click here to connect to the Microsoft Solution Provider Program page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Microsoft Solution Providers (SPs) are independent organizations that have teamed with Microsoft to use technology to solve business problems for companies of all sizes and industries. SPs use the Microsoft Solutions Platform of products as building blocks and offer various value-added services, such as integration, consulting, software customization, developing turn-key applications, and technical training and support. All Solution Providers have at least one Microsoft Certified Professional on staff who has demonstrated expertise in developing, implementing, and supporting Microsoft solutions.

Microsoft SP's understand how to help organizations take advantage of today's powerful new graphical client-server applications to make sound business decisions. By partnering with Microsoft, Solution Providers can combine their own high-quality services and expertise with Microsoft's powerful technologies, products and information.

The Microsoft Solution Provider team is a full-service commitment to your organization. We offer the following advantages:

Understanding of Your Business

Solution Providers — working with Microsoft — have the knowledge and expertise to match technology to the exact needs of any business. Some SPs focus on a specific vertical market; others serve organizations in a wide range of industries. All SPs are carefully screened and qualified to ensure consistent, high-quality capabilities.

World-Wide Diversity

The Microsoft Solution Provider program includes thousands of Solution Providers (and tens of thousands of technical experts) throughout the world. Often, multiple Solution Providers, each with a specific area of expertise, will team up to provide a comprehensive solution to a customer's needs.

Responsive, Wide-Ranging Service

Solution Providers offer a level of expertise, support, and training not always available or affordable within a single organization, with services ranging from computer network design consulting to installation and highly specialized custom application development, to on-site training and support. Many SPs can also integrate and develop solutions using multiple vendors' products and tools.

The Solution Provider Exchange Specialist Initiative

One of the benefits offered through the Solution Provider program is the ability to participate in various "specialist" programs. As a Solution Provider Exchange Specialist, you will be given technical training, resources, and customer leads. To qualify, your organization must be an active member of the Solution Provider program, and technical staff must pass an examination on Microsoft Exchange Server.

Other Microsoft Solution Provider Specialist Programs include: Windows 95 and Office for Windows 95 Migration Specialist, Internet Specialist, and 32-bit Migration Specialist.

Enrolling in the Solution Provider Program

To obtain an application to become a Microsoft Solution Provider in the United States and Canada, call Microsoft toll-free at: (800) SOLPROV [(800) 765-7768].

Outside the U.S. or Canada, please contact your local subsidiary. Customers who are deaf or hard of hearing can reach Microsoft text telephone (TT/TDD) services by calling (800) 892-5234 in the United States or (905) 568-9641 in Canada.

Click here to connect to the Support Online page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Microsoft offers technical support and services, ranging from no-cost and low-cost online information services (available 24 hours a day, 7 days a week), to annual support plans. The Microsoft Support Online Web site contains the following sources of support:

® "How to" information

® Solutions to problems

® Drivers, patches, and code samples

® Answers to frequently asked questions

® Opportunities to share information and expertise with other customers

® Information about other service offerings

The *Mastering Web Site Development* Help file also contains information about Microsoft Technical Support. See the last topic in the Help file, "When You Need Technical Support."

Microsoft provides a variety of training options for developers, including classroom training at Microsoft Authorized Technical Education Centers (ATECs), online training, and self-paced training such as this and other Mastering Series titles.

® Face-to-Face Training

® Microsoft Online Institute

® Self-Paced Training

Microsoft Official Curriculum courses are available through Microsoft authorized training sites. Microsoft Solution Provider Authorized Technical Education Centers (ATECs) are commercial training organizations offering Microsoft Official Curriculum over consecutive days, and Microsoft Authorized Academic Training Program (AATP) institutions offer Microsoft Official Curriculum over an academic term. These professional education centers deliver consistent, high-level, hands-on technical training on the full range of Microsoft productivity, networking, operating system, and application development products. All Solution Provider ATECs and AATP sites have met stringent guidelines to receive their Microsoft authorization. These guidelines govern curriculum, class sizes, training facilities, and, most importantly, the educational and technical expertise of the Microsoft Certified Trainers. This means you receive the highest level of curriculum and education services available in the industry. For example:

- ® Only Microsoft authorized training sites may use Microsoft Certified Trainers to deliver Microsoft Official Curriculum.
- ® Microsoft Certified Trainers undergo rigorous training and testing by Microsoft to become certified to teach Microsoft Official Curriculum. They are experts in providing training on Microsoft operating systems, the Microsoft BackOffice family of server software, and Microsoft desktop and development tools.
- ® Solution Provider ATECs and AATPs receive early releases of Microsoft software and course materials so they can begin classroom instruction as soon as Microsoft products are released to the public.

For full course descriptions and a referral to a Microsoft Solution Provider Authorized Technical Education Center, call (800) SOLPROV in the United States and Canada. In other countries, contact your local Microsoft office.

Ask for the Roadmap to Microsoft Education and Certification, an online windows-based information tool. Or see E&CMAP.ZIP from Library 5 of the Solution Provider forum on CompuServe (GO MSEDCERT).

Check out the details in the section titled Microsoft Certified Professional Program.

Click here to connect to the Microsoft Online Institute campus on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The Microsoft Online Institute (MOLI) is an online learning environment that brings in-depth Microsoft technical training directly to your desktop. Utilizing the latest in Internet technology, the Microsoft Online Institute pairs self-study learning materials with highly-qualified instructors, providing an effective, flexible, and low-cost training solution for busy professionals.

The Microsoft Online Institute offers a growing list of classes to help you gain the latest knowledge and skills on Microsoft products and technologies, and to help you pursue your Microsoft Certified Professional goals.

Best of all, you can progress at your own pace, on your own schedule. Available 24 hours a day, 7 days a week, the Microsoft Online Institute gives you the training you need when and where you need it.

And when you need help, your instructor will be available online. Take advantage of instructors with "real world" experience who will put a practical spin on course material. Send your questions via e-mail and get a personal reply. Join scheduled real-time chat sessions with your instructor and other students.

By attending a virtual classroom on the Web, you stay in touch with a real instructor and keep motivated to move along with the course materials.

Microsoft Official Curriculum is also available in self-paced training formats for those who prefer to learn on their own, at their own pace. These materials are available in book, computer-based training (CBT), and mixed-media (book and video) formats. A Solution Provider ATEC in your area can provide you with self-paced training materials, or they can be purchased wherever Microsoft Press books are sold.

Additional self-paced training materials are developed by third-party companies partnering with Microsoft. Microsoft Certified Professional (MCP) Approved Study Guides have been carefully and rigorously reviewed and approved by Microsoft as an effective way to prepare for Microsoft certification. Look for the Microsoft Certified Professional Logo (see Microsoft Logo Program) prominently displayed on all MCP Approved Study Guides, found at retail bookstores.

The next section provides details on all of the currently available Mastering Series titles:

® The Microsoft Mastering Series

Click here to connect to the Mastering Series page on the Microsoft Web site. Use this web site to get up-to-date information on new and upcoming releases.
{ewc mvimg, mvimage,!intjump.bmp}

The Microsoft Mastering Series titles are content-rich, self-paced, interactive CD-ROM–based learning tools that are designed to help you quickly master Microsoft development tools. The courses combine high-level technical content with an intuitive learning methodology, so you can learn in the way that best suits your schedule and study habits.

Mastering Web Site Development is the latest in the Mastering Series of titles. The other Mastering Series titles currently available are:

® Mastering Microsoft Visual FoxPro

® Mastering Microsoft Exchange Development

® Mastering Internet Development

® Mastering Microsoft Office 97 Development

® Mastering Microsoft Visual Basic 5

® Mastering MFC Development Using Visual C++ 5

® Mastering Visual J++

All Mastering titles include interactive lessons, narrated demonstrations, hands-on labs, sample code, and a library of reference material to supplement courses and exercises. A flexible browsing and searching system allows you to navigate directly to the course material or library resources that you need.

Most of the Mastering Series courses are Microsoft Official Curriculum and can be used to gain the skills necessary for Microsoft Certification. Microsoft Mastering Series products are not introductions to programming. It is assumed that you have experience using Microsoft development tools to create solutions for Microsoft Windows.

Course No. 676

This course is intended for developers and system administrators responsible for implementing applications for Microsoft Exchange.

This course teaches developers to design and build sophisticated workgroup applications around Microsoft Exchange using the Microsoft Visual Basic programming system. It also teaches nonprogrammers, such as system administrators, to build simpler form and folder applications on Microsoft Exchange without programming.

The topics covered in this course are:

® Designing a Microsoft Exchange application

® Implementing public folders

® Creating Microsoft Exchange Forms Designer applications

® Using Visual Basic to extend a form

® OLE Messaging

® OLE scheduling

® Intergrating Microsoft Office and Microsoft Exchange

At Course Completion

At the end of the course, students will be able to design a Microsoft Exchange application, implement public folders, create Forms Designer applications, extend form applications using Visual Basic, use OLE messaging, use OLE scheduling, and integrate Microsoft Exchange applications with Microsoft Office.

Microsoft Certified Professional Exams

Information on Microsoft Certified Professional exam preparation provided by this course is yet to be decided.

Prerequisites

Students should have a basic understanding of using an electronic mail system.

To create basic form and folder applications, students do not need any programming experience.

To complete the more advanced sections of the course, students should have intermediate experience using Visual Basic. Students should be able to:

® Create an application in Visual Basic that uses multiple forms.

® Use OLE Automation to control other applications.

® Retrieve and update database information using data access objects (DAOs).

Chapter 1: Designing Solutions

Topics

® Development overview

® Development tools

® Creating solutions

Skills

Students will be able to:

® Describe the major components of Microsoft Exchange.

® Describe the types of solutions that you can build using Microsoft Exchange.

® Describe the Microsoft Exchange development tools.

® Choose the right tools to build a Microsoft Exchange solution.

® Describe the difference between a Microsoft Exchange form application and an external mail-enabled application.

Chapter 2: Implementing Folders

Topics

® Creating folders

® Using forms in folders

® Creating views

® Folder permissions

® Setting rules

® Using a folder offline

® Replicating folders

® Managing folders

Lab

Implementing folders

Skills

Students will be able to:

® Create a new public folder or copy an existing folder.

® Associate forms with a folder and remove forms from a folder.

® Define views for a folder.

® Grant permissions on a public folder.

® Set rules for a folder.

® Make a folder available for use offline and synchronize an offline folder.

® Explain the fundamentals of replication.

® Manage folders.

Chapter 3: Creating Basic Form Applications

Topics

® Forms Designer

® Creating forms

® Creating fields

® Saving and installing forms

Lab

Creating basic form applications

Skills

Students will be able to:

® Use Forms Designer to create a basic Post and Send form.

® Install a Microsoft Exchange form.

® Differentiate between global forms, private forms, and forms associated with a folder.

® Differentiate between a Post form and a Send form.

® Create a form that enables a user to include an attachment.

® Perform form administration tasks such as determining which forms are associated with a folder, copying, moving, deleting, and adding forms to folders.

® Describe how forms are registered and activated.

Chapter 4: Creating Advanced Form Applications

Topics

® Adding a second window to a form

® Using custom response events

® Adding Help information

Lab

Creating advanced form applications

Skills

Students will be able to:

® Differentiate between a window and a form.

® Create a form that contains a Read window and a Compose window.

® Create a form with multiple custom responses.

® Share fields between forms.

® Add Help information to a form, a window, and fields.

® Modify standard menus on a form.

® Create an application that consists of multiple forms.

Chapter 5: Extending Forms: Basic Modifications

Topics

® Understanding the Visual Basic–based project

® Common modifications

® Testing your form

Lab

Extending forms: Basic modifications

Skills

Students will be able to:

® Determine when you need to extend a form generated by Forms Designer.

® List the files generated by Forms Designer.

® List the typical files and routines that are modified to extend a form application.

® Use Visual Basic to modify a project created by Forms Designer.

® Recompile and reinstall a modified form.

® Define the purpose of a .CFG file.

Chapter 6: Extending Forms: Advanced Modifications

Topics

® Accessing databases

® Adding controls

® Creating a routing form

® Adding a file

Lab

Extending forms: Advanced modifications

Skills

Students will be able to:

® Extend a form to read and write database information.

® Add new controls to a form.

® Add routing features to a form.

® Specify files, such as a Help file, to install with your form.

Chapter 7: Using OLE Messaging

Topics

® Creating a messaging application programming interface (MAPI) session

® Working with messages

® Working with addresses

® Working with folders

Lab

Using OLE messaging

Skills

Students will be able to:

® Use basic OLE Automation.

® Describe how OLE messaging is used.

® Describe the OLE messaging object model.

® Create a Visual Basic–based application that:

- Uses OLE messaging to send a message with and without a Compose dialog box.
- Uses OLE messaging to read messages.
- Sends and retrieves an attachment in a message.
- Searches for a specific message.
- Adds custom properties to a message.
- Reads custom properties from a message.
- Deletes a message.
- Moves a message from one folder to another folder.
- Uses the personal address book to address a message.
- Sends a message to a public folder.

Chapter 8: Using OLE Scheduling

Topics

® Working with appointments

® Working with projects and tasks

® Working with contacts

® Distributing an application

Lab

Using OLE scheduling

Skills

Students will be able to:

® Describe the OLE scheduling object model.

® Create a Visual Basic–based application that:

- Creates and views an appointment.
- Creates and views a task.
- Adds and views a contact.
- Reads all appointments.

Chapter 9: Integrating Microsoft Exchange and Microsoft Office

Topics

® Posting Office documents

® Sending and routing documents

® Office features

Lab

Integrating Microsoft Exchange and Microsoft Office

Skills

Students will be able to:

® Send, route, and post documents manually or programmatically from within the Microsoft Office programs.

® Create custom document properties.

® Create a linked custom property.

® Group and sort on custom properties.

® Describe how to use Microsoft Word as an e-mail editor.

® Describe how your personal address book can be used in a Word mail merge.

® Describe how project scheduling can be integrated with Schedule+.

Course No. 625

This course is intended for experienced database developers who create and distribute applications.

Microsoft Mastering Series titles are available through major bookstores, software resellers, and through Microsoft Authorized Technical Education Centers (ATECs) For a referral to an ATEC in your area, call (800) SOLPROV.

Delivered on CD-ROM, this course is the first in a series supporting developers who create mission-critical applications. Focusing on the newest release of the Microsoft Visual FoxPro database management system (DBMS), this course offers in-depth interactive training for experienced developers.

At Course Completion

At the end of the course, students will be able to utilize the full functionality of object-oriented programming; use the core set of commands and functions in object-oriented programming; apply the hierarchy of the classing structure; design and create additional functionality for forms; add functionality to the grid; consider multiuser issues during the application development process; manage and control data in multiuser situations; use client/server support to work with remote data; use OLE objects with Visual FoxPro version 3.0 and Microsoft Office products; and apply basic conceptual architecture when creating wizards.

Microsoft Certified Professional Exams

This course helps you prepare for the Microsoft Certified Trainer exam for Visual FoxPro.

Prerequisites

® Experience working with the Microsoft Windows operating system

® Knowledge of relational database design concepts

® Fundamental programming skills

® The ability to apply the fundamentals described in the Visual FoxPro User's Guide

The course materials, lectures, and lab exercises are in English. To benefit fully from our instruction, students need an understanding of the English language and completion of the prerequisites.

Course Materials and Software

This course is licensed for use by a single user on a single computer. Multiuser usage is prohibited. All content on the CD-ROM can be printed.

Module 1: Project Manager

Topics

® Overview

® Creating applications

® Converting FoxPro 2.*x* project files

® Converting catalog files

Skills

Students will be able to:

Use Project Manager to build new applications, manage project files, and convert existing project or catalog files.

Module 2: Wizards and Builders

Topics

® Overview
® Wizards
® Builders
® Architecture

Skills

Students will be able to:

® Use Wizards to accomplish tasks.
® Use Builders to add controls to forms.
® Understand architecture of wizards and builders.

Module 3: Form Designer

Topics

® Overview
® Designing forms
® Form controls
® Data environment
® Data sessions
® Form sets
® Setting properties
® Working with the grid
® Page frames
® OLE Controls

Labs

® Working with grid positioning
® Manipulating the grid
® Adding a control programmatically
® Adding a control visually
® Using the sparse property
® Calling a method programmatically
® Changing grid color and bringing up a browse
® Changing column colors and reset
® Changing the color of the active cell
® Common tasks
® Creating a form set that updates and skips through records
® Creating a one-to-many form
® Data environment
® Drag and Drop
® Grid cells with different colors
® Multiple sessions
® OLE Controls
® Page frames
® User-defined controls

® Working with a grid through code

Skills

Students will be able to:

® Use Form Designer and available controls.

® Use Data Environment Designer.

® Create form sets.

® Set properties.

® Manage controls and user interfaces with page frames.

® Display and manipulate data with the Grid tool.

® Understand the benefits of OLE Controls.

Module 4: Query Designer

Topics

® Overview

® Query Designer

Skills

Students will be able to:

® Use Query Designer to extract records.

® Specify criteria to extract records.

Module 5: Report Designer

Topics

® Overview

® Accessing Report Designer

® User interface

® Ease-of-use features

Skills

Students will be able to:

Use Report Designer to summarize, format, and print data.

Module 6: Error Management

Topics

® Overview

® Types of errors

® Debugging tools

® Error handling

Labs

® Checking values in the debug window

® Creating an error-handling routine

Skills

Students will be able to:

® Use command, debug, and trace windows to resolve errors.

® Use an error-handling routine to resolve run-time errors.

Module 7: Object Model

Topics

® Overview

® What is object-oriented technology?

® Terminology

® Properties, events, and methods

® Scoping of variables

® Object-oriented language

Labs

® Using the (::) scope resolution operator

® ParentClass property

® Creating a form programmatically

® Using the DEFINE CLASS command to create a subclass

® Adding controls to a form using the AddObject() method

® Creating a custom function

Skills

Students will be able to:

® Understand benefits of object-oriented programming.

® Understand hierarchical structure and related terminology.

® Define properties.

® Scope variables.

® Work with the Visual FoxPro language.

Module 8: Visual Classes

Topics

® Overview

® What are classes?

® Types of classes

® Creating classes and subclasses

® Class browser

Labs

® Creating a form class

® Creating and registering a button class

® Subclassing from an existing class

® Modifying an existing class

® Creating a container class

® Creating a control class

® Creating a custom class

® Toolbar class

Skills

Students will be able to:

® Understand benefits of classes.

® Understand types and hierarchies of classes.

® Create classes and subclasses.

® Use the Class browser.

Module 9: OLE Automation

Topics

® Overview

® What is OLE?

® OLE features

® Using OLE Automation objects

® Controlling Microsoft Excel from Visual FoxPro

® Controlling Microsoft Word from Visual FoxPro

Skills

Students will be able to:

® Understand benefits of OLE.

® Understand how Visual FoxPro stores and converts OLE objects.

® Understand the OLE features in Visual FoxPro.

® Use OLE Automation to create and manipulate objects.

® Control Microsoft Excel and Word.

Module 10: Shared Access to Data

Topics

® Overview

® Controlling multiuser access

® Buffering data

® Managing data and conflicts

® Transaction support

Skills

Students will be able to:

® Control access to data.

® Buffer data.

® Manage conflicts.

® Use transactions to maintain data integrity.

Module 11: Client/Server

Topics

® Overview

® Establishing a connection

® SQL pass-through connections

® What is a view?

® Local and remote views

® Fetching data

® Updating, deleting, and inserting data on a remote server

Skills

Students will be able to:

® Connect to a remote server using SQL pass-through connections.

® Retrieve an updatable result set using View Designer.

® Explain local and remote views.

® Fetch data.

® Manipulate and update data on a remote server.

® Use Upsizing Wizard.

Module 12: Dynamic-Link Libraries (DLLs)

Topics

® Overview

® What DLL procedures can I call?

® Using DLLs

Labs

® Nulls

® Passing arguments by value or by reference

® Windows directory (listed under Labs for the Table of Contents but not in the course chapter)

® Using an alias for application programming interface (API) functions

® Handles

® Hiding the main FoxPro window

® Using a form to hide the main FoxPro window

Skills

Students will be able to:

® Use DLLs.

® Know sources of DLLs.

Course No. 677

Mastering Internet Development with Microsoft ActiveX Technologies is intended for content developers who are responsible for creating Web pages, and programmers who want to make their existing Visual Basic–based and Visual C++–based applications Internet-aware.

This course will teach developers how to publish content on the Internet and create applications that connect with the Internet by focusing on authoring active Web content and developing Internet-aware applications.

At Course Completion

At the end of the course, students will be able to install an Internet server with Microsoft Internet Information Server, and author a Hypertext Markup Language (HTML) page (create static content) using Internet Assistant for Microsoft Word for Windows 95. They will also be able to add code to an HTML page (create active content) using Microsoft ActiveX controls and the Visual Basic programming system, Scripting Edition, and use the Microsoft Internet Explorer object model from a Visual Basic–based or Visual C++–based application. In addition, students will be able to read and write information to an Open Database Connectivity (ODBC) database from a Web page using the Internet server application programming interface (ISAPI) extension provided with Internet Information Server, and call a method of an OLE Automation server from a Web page using the ISAPI extension. Finally, students will be able to write their own ISAPI server extensions and filters using the Visual C++ development system.

Microsoft Certified Professional Exams

This course helps you prepare for the following Microsoft Certified Professional exams:

® To be determined

Prerequisites

This course assumes basic experience installing the Microsoft Windows NT Server network operating system, using the Visual Basic or Visual C++ to create applications for Windows, and using Word to create a document. This course does not assume any prior experience or knowledge of the Internet.

Potential students should be able to accomplish the following tasks before taking this training:

® Install Windows NT Workstation and Windows NT Server.

® Set up security for Windows NT Server.

® Install Transmission Control Protocol/ Internet Protocol (TCP/IP).

® Set up a system ODBC data source.

Content Developers:

® Compose a new document based on a template.

® Open a document of one type and save it in a different format.

® Create a table in a document.

® Insert a graphic into a document.

Programmers:

® Use Visual Basic to add controls to a form, place code in the appropriate events, and create an executable.

or-

® Use Visual C++ and wizards to create a Single Document Interface (SDI) application.

® Add ActiveX controls to a Visual Basic–based or Visual C++–based application.

The course materials are in English. To benefit fully from the course, students need an understanding of the English language and completion of the prerequisites.

Course Materials and Software

This course is licensed for use by a single user on a single computer. Multiuser usage is prohibited. All content on the CD-ROM can be printed.

Module 1: Architecture of the Internet

Topics

® History

® Architecture of the Internet

® Intranet versus Internet (including limitations of intranet structure)

® Using person-to-resource and person-to-person services on the Internet

® Using Uniform Resource Locators (URLs)

® Introduction to the World Wide Web

Labs

® Exploring Internet services

® Using URLs on the Web

Skills

Students will be able to:

® Describe why protocols are important for communicating on the Internet.

® Describe the function of the Internet backbone.

® Define a URL.

® Discriminate between the Internet and an intranet.

® Name three Internet services.

® Define the role of an Internet service provider.

® Describe the purpose of a “newsgroup.”

® Describe the purpose of HTML.

® Discriminate between an active and a static Web page.

® Use FTP to download files.

® Search the Web for a given topic.

® Add a URL to a Windows-based application.

Module 2: Installing an Internet Server

Topics

® Overview

® Microsoft Internet Information Server

® Extending the server

Labs

® Setting up Microsoft Internet Information Server

® Defining security for your Web server

Skills

Students will be able to:

® Explain the reasons for setting up an Internet server.

® Install Microsoft Internet Information Server.

® Configure Web services on Internet Information Server with Information Server Manager.

® Add security to an Internet server.

Module 3: Authoring a Static Web Page

Topics

® Designing and managing a Web site

® Introduction to HTML

® Creating an HTML page with Internet Assistant for Word

® Converting an existing document to HTML format

® Adding advanced features such as frames, image maps, and server-side includes to an HTML page

Labs

® Creating an HTML page

® Converting an existing document to HTML format

® Adding links

® Using frames, image maps, and server-side includes

Skills

Students will be able to:

® List the elements that make up the basic structure of an HTML file.

® Compare and contrast the Microsoft authoring tools that are available and select the appropriate tool for a given task.

® Use Internet Assistant for Microsoft Word, Microsoft Excel, or Microsoft Access to convert an existing document to HTML format.

® Using Internet Assistant for Word, create a new static HTML page that incorporates formatted text, links, graphics, and other special effects.

® Add Microsoft Internet Explorer–specific tags to an HTML page to expose features not supported by Internet Assistant for Word.

® Explain why server-side includes are useful.

® Use a server-side include, image maps, and frames on a Web page.

Module 4: Adding Controls and Other Objects to a Web Page

Topics

® Overview

® Adding standard controls with Internet Assistant for Word

® Adding ActiveX controls and Java “applets”

Labs

® Adding standard HTML controls

® Adding ActiveX controls

Skills

Students will be able to:

® Explain the difference between a static and an active Web page, and explain the role of controls in creating active Web content.

® Explain the purpose of an HTML form in an HTML page.

® Add a standard control to an HTML form.

® Add an ActiveX control to an HTML page.

Module 5: Adding Client-side Scripts with Visual Basic, Scripting Edition

Topics

® Overview of client-side scripting

® What is Visual Basic, Scripting Edition?

® Controlling controls with Visual Basic, Scripting Edition

® Using the Microsoft Internet Explorer object model for scripting

Labs

® Writing control event procedures

® Communicating with an ActiveX control on the client

® Writing validation code

Skills

Students will be able to:

® Create an event procedure for a standard control using Visual Basic, Scripting Edition.

® Create an event procedure for an ActiveX Control using Visual Basic, Scripting Edition.

® Control Microsoft Internet Explorer from a Web page.

® Add page initialization code to a Web page with Visual Basic, Scripting Edition.

Module 6: Communicating with the Server

Topics

® Overview

® Linking to an application on the server

® Sending information to the server with a form

® Using server-extension sample applications

® Calling an OLE Automation server from a Web page

Labs

® Using the FormDump server extension

® Using the ReDir server extension

® Accessing an OLE Automation server

Skills

Students will be able to:

® Explain the process by which a Web page communicates and interacts with the Internet server.

® Create an HTML page that is able to send information from a form to the server.

® Explain the role of the OLE Automation server in creating active content on the Web.

® Create an HTML page that can communicate with an OLE Automation server located on the Internet server.

Module 7: Enabling Data Access Using ODBC

Topics

® Overview of the Internet Database Connector server extension

® Constructing a simple query

® Constructing a query with parameters

® Writing information from the user into a database on the server

Labs

® Reading information from an ODBC database

® Passing parameters to the database query using a URL

® Passing parameters to the database query using a form

® Returning links from the database

® Writing information into the ODBC database

Skills

Students will be able to:

® Explain the role of the ODBC database in creating active content on the Web.

® Create an HTML page that can read and write information to an ODBC data source located on the server.

Module 8: Automating Microsoft Internet Explorer

Topics

® Applications and the Internet

® Microsoft Internet Explorer 3.0 object model

® Creating an instance of Microsoft Internet Explorer from a Visual Basic–based application

® Using the Microsoft Internet Explorer control from a Visual Basic–based application

® Working with the Microsoft Internet Explorer objects

Labs

® Automating Microsoft Internet Explorer

® Using the WebBrowser control

® Downloading a file

Skills

Students will be able to:

® Discuss the implications of creating an Internet-aware application, and list some features typically found in this type of application.

® Using Visual Basic, create an application that views HTML pages on the Web.

Module 9: Mail-enabling an Application

Topics

® Overview of mail technologies

® Adding simple messaging application programming interface (Simple MAPI) to a Visual Basic–based or Visual C++–based application

® Adding Extended MAPI to an application using OLE messaging

® Working with the OLE messaging objects

Lab

® Using MAPI to mail-enable an application

Skill

Students will be able to:

® Create a Visual Basic–based or Visual C++–based application that sends and receives mail through MAPI.

Module 10: Developing Server-side Extensions

Topics

® Overview

® Building an ISAPI filter

® Building an ISAPI application

Labs

® Creating an ISAPI filter

® Creating an ISAPI application (dynamic-link library)

Skills

Students will be able to:

® Describe the role of server-side Internet components.

® Explain how server-side preprocessing works.

® Explain when ISAPI filters are called.

® Create a simple ISAPI filter.

® Create an ISAPI application (DLL).

Course No. 778

This course is intended for corporate developers and solution providers who need to create and distribute custom solutions developed with Microsoft Office.

This course syllabus should be used to determine whether the course is appropriate for the student, based on current skills and technical training needs. Technical information is provided on the intended audience, course prerequisites, covered topics, lab exercises, course materials, and software.

Course content, prices, and availability are subject to change without notice.

The primary objective of this course is to teach students how to develop custom solutions that integrate components of the Microsoft Office 97 family.

At Course Completion

At the end of the course, students will be able to design a custom solution; use the Microsoft Visual Basic programming system environment in each of the Microsoft Office applications; use Visual Basic for Applications to write code; describe the purpose of object models and use automation to program applications; list and describe the object models provided in the Office Developer Edition; customize the user interface of Office applications; use data access objects (DAOs) and ODBCDirect in an Office application to programmatically access external data; add intranet links to Office applications and use the Office Web object model; add workgroup features to custom solutions; and distribute a custom solution.

Microsoft Certified Professional Exams

This course helps you prepare for the following Microsoft Certified Professional exams:

® To be determined

Prerequisites

® Be able to use basic features of the Microsoft Office applications

® Understand event-driven programming concepts

® Understand basic database concepts such as tables and queries

The course materials are in English. To benefit fully from the course, students need an understanding of the English language and completion of the prerequisites.

Course Materials and Software

The course CD-ROM is yours to keep. This course is licensed for use by a single user on a single computer. Multiuser usage is prohibited. All content on the CD can be printed.

You will be provided with the following software for use in the classroom:

® Microsoft Windows 95 operating system

® Microsoft Office 97 Developer Edition

Chapter 1: Designing a Custom Solution

Topics

® Microsoft Office 97 development features

® Design process

® Office 97 applications

® Developing with Office 97

® Sample custom solutions

Lab
Using the sample solution

Skills

Students will be able to:
® Determine the Office 97 application best suited to solve a business need.

® Describe the developer features found in Office 97.

® Describe the solution architecture of a custom solution.

® Describe where you can write code for your custom solution.

Chapter 2: Using the Visual Basic Editor

Topics
® Where code is stored

® Working with the Visual Basic editor

® Writing code

® Running code

® Reusing code

Lab
Using the Visual Basic editor

Skills

Students will be able to:
® Describe how projects relate to documents.

® List the components that you can include in a project.

® List where code is stored.

® Record code in Microsoft Excel, Word, and the PowerPoint presentation graphics program.

® Use the Visual Basic editor to view, modify, and run code.

® Assign code to a menu or a command button on a toolbar.

Chapter 3: Using Visual Basic for Applications

Topics
® Variables

® Procedures

® Controlling program execution

® Errors

® Debugging

Lab
Using Visual Basic for Applications

Skills

Students will be able to:
® Declare and use variables.

® Determine the appropriate data type to use in a particular scenario.

® Create and invoke *Sub* and *Function* procedures.

® Use decision control and looping statements to control program execution.

® Add error-handling statements to handle run-time errors.

Chapter 4: Using Forms and Controls

Topics

® Using controls on forms

® Working with forms

® Using controls on documents

Lab

Using controls

Skills

Students will be able to:

® Create and use user forms in Microsoft Excel, Word, and PowerPoint.

® Add controls to a Microsoft Office document.

® Link the contents of controls to worksheet ranges in Microsoft Excel.

Chapter 5: Object Models and Automation

Topics

® Navigating an object hierarchy

® Working with objects

® Automating Microsoft Office applications

Lab

Object models and automation

Skills

Students will be able to:

® Describe the purpose and benefits of using an object model.

® Use the object browser to view an object model.

® Describe the difference between objects, properties, and methods.

® Write code with Visual Basic for Applications to automate applications.

® List and describe the object models provided in Microsoft Office Developer Edition.

Chapter 6: Using Microsoft Excel Objects

Topics

® Using workbooks

® Using worksheets

® Creating charts

® Creating PivotTable dynamic views

Lab
Using Microsoft Excel objects

Skills
Students will be able to:
® Programmatically create and modify workbooks and worksheets.
® Programmatically add values and formulas to a worksheet.
® Write code to workbook and worksheet events.
® Programmatically create and modify a chart.
® Programmatically create and modify a PivotTable.

Chapter 7: Using Word Objects

Topics
® Word objects
® Navigating the Word object model
® Using events in Word
® Working with Word documents
® Working with areas of a document
® Working with text
® Working with Word forms
® Using Mail Merge

Lab
Using Word objects

Skills
Students will be able to programmatically:
® Create and modify Word documents.
® Create a document based on a Word template.
® Format text in a Word document.
® Create and use Word forms.
® Use Mail Merge.

Chapter 8: Using PowerPoint Objects

Topics
® Working with presentations
® Working with slides
® Working with a slide show

Lab
Using PowerPoint objects

Skills
Students will be able to:

® Create and modify a PowerPoint presentation.

® Add shapes to a slide.

® Use the *OnAction* property to determine how a shape responds to mouse actions during a slide show.

® Add speaker notes to a slide.

® Modify the master slide in a presentation.

® Create a custom show.

Chapter 9: Using Microsoft Access Objects

Topics

® Introduction to programming with Microsoft Access

® Using forms

® Using reports

Lab

Using Microsoft Access objects

Skills

Students will be able to:

® Run a Microsoft Access form or report from another Office 97 application.

® Write code for a form, report, and control events in Microsoft Access.

® Programmatically create forms and reports in Microsoft Access.

® Programmatically add a code module or code to an existing code module for a form or report in Microsoft Access.

® Programmatically add controls to a form or report in Microsoft Access.

Chapter 10: Shared Object Models

Topics

® Customizing menus and toolbars

® Programming the Assistant

® Searching for files

® Drawing objects

Lab

Shared object models

Skills

Students will be able to:

® Use the CommandBars object model to modify and create menus and toolbars.

® Use the Assistant object model to retrieve and display information.

® Use the FileSearch object model to search for files.

® Use the Drawing object model to create graphics.

Chapter 11: Using DAOs

Topics

® Microsoft Jet databases

® ODBCDirect Workspace

Lab

Using data access objects

Skills

Students will be able to use DAOs to:

® Open a Microsoft Jet database.

® Retrieve and update records.

® Create and execute a query.

® Refresh a linked table.

® Open a connection to an Open Database Connectivity (ODBC)–compliant data source.

® Retrieve records from an ODBC-compliant data source.

Chapter 12: Using Outlook Objects

Topics

® Designing Outlook forms

® Automating with Outlook

Lab

Using Outlook

Skills

Students will be able to:

® Create, publish, and use Outlook forms.

® Add code to an Outlook form.

® Use the Outlook object model programmatically to create and use electronic-mail messages, contacts, tasks, and appointments.

Chapter 13: Adding Intranet Features

Topics

® Working with hyperlinks

® Saving documents in hypertext markup language (HTML) format

® Using the WebBrowser control

Lab

Adding intranet features

Skills

Students will be able to:

® Describe the difference between the Internet and an intranet.

® Add hyperlinks to a document.

® List the advantages of Microsoft ActiveX technology.

® Save documents in HTML format.

® Use the WebBrowser control on a document or form.

Chapter 14: Distributing a Custom Solution

Topics

® Securing your custom solution

® Working with add-ins

® Using references

® Using Setup Wizard

Skills

Students will be able to:

® Set a password for a document, database, and Visual Basic for Applications project.

® Understand the security options in Microsoft Access.

® Load and unload an add-in in Microsoft Excel, Word, PowerPoint, and Microsoft Access.

® Describe the differences among add-ins in Microsoft Excel, Word, PowerPoint, and Microsoft Access.

® Set a reference to an Office document manually and programmatically.

® Use Setup Wizard in Office Developer Edition to create a Setup program for your custom solution.

Course No. 789

Because some parts of the course are currently being developed, some elements of this syllabus are subject to change.

This course is intended for developers, systems analysts, and information managers who are responsible for implementing sophisticated C++ applications using Microsoft Foundation Classes, Active Template Library, and Component Object Model technologies.

This course syllabus should be used to determine whether the course is appropriate for the student, based on current skills and technical training needs. Technical information is provided on the intended audience, course prerequisites, covered topics, lab exercises, course materials, and software.

Course content, prices, and availability are subject to change without notice.

The primary goal of this course is to teach students how to develop applications and components for database front ends and the Internet using the Microsoft Visual C++ development system, Microsoft Foundation Classes (MFC), Active Template Library, and Component Object Model (COM) technologies.

At Course Completion

At the end of the course, students will be able to describe the types of applications and components that can be developed using Visual C++; describe the basic architecture of an MFC application and the general structure of the MFC Library; debug MFC applications; use MFC to design, create, and implement menus, toolbars, status bars, and dialog bars; design, build, and test dialog boxes; describe the view classes available in MFC and show how to implement them, case by case, in an application; build database applications; create custom queries, select specific records, and lock records using MFC; use the Web Browser control, WinInet, and WinSock classes; build MFC Internet applications; create ActiveX components using the ActiveX control wizard and use them in an MFC application; create COM objects using ATL COM AppWizard, and use them in an MFC application; and create Internet Server Application Programming Interface (ISAPI) extensions for both applications and filters.

Prerequisites

This course assumes the user has experience and knowledge in the following areas:

® Basic C++ programming skills

® All C++ coding constructs common with C

® Inheritance, polymorphism, overloading, default arguments

® Some programming experience with MFC Library

® Microsoft Windows–based user skills, including using a mouse, menus, the file system, and the Microsoft Internet Explorer interface

® Windows architecture concepts, including defining event-driven programming, processes, virtual memory models, threading, multitasking, and messaging

Experience using a bitmap editor, a dialog editor, and the Visual C++ menu editor

Familiarity with object-oriented programming terminology and concepts such as objects, properties, and methods

The course materials are in English. To benefit fully from the course, students need an understanding of the English language and completion of the prerequisites.

Course Materials and Software

This course is licensed for use by a single user on a single computer. Multiuser usage is prohibited. All content on the CD-ROM can be printed.

The CD-ROM includes:

® Course chapters incorporating text, sound, video, graphics, animation, tips, cross-references to related

information, and self-check questions

® Extensive lab exercises

® Sample code

® Library of related technical articles

® Index and glossary system for accessing information via keywords

® Help

Chapter 1: MFC and Component Technology

Topics

® Advantages of Microsoft Foundation Classes, Visual C++, and ATL

® What's new in MFC, Visual C++ and the Microsoft Developer Studio

® Special features provided by MFC

® COM, and Active Template Library

® What kinds of applications and components you will be able to build with Visual C++

Skills

Students will be able to:

® List the types of applications and components one can build with Visual C++.

® List the special features of those applications.

® Explain the advantages MFC, ATL and Visual C++ offer to the developer.

Chapter 2: MFC Architecture

Topics

® Classifications of MFC

® Document/view architecture

® Messaging and message handling

® Registry support in MFC

® Non-document/view architecture

Skills

Students will be able to:

® Describe the major categories of classes in MFC Library.

® Describe the document/view architecture.

® Describe the structure and main components of an MFC-based application.

® Describe MFC support for the registry.

Chapter 3: Using Developer Studio

Topics

® Microsoft Developer Studio

® Wizards

® Developer Studio Gallery

® Resource editors

® Information resources for Visual C++

Labs

® Creating an MFC application

Skill

Students will be able to:

® Use Developer Studio to create and view information about MFC applications.

Chapter 4: Debugging and Error-Handling

Topics

® The Debugging Environment

® Using Developer Studio Debugger

® Adding MFC debugger support

® Handling errors and exceptions

® Debugging special cases

Skills

Students will be able to:

® Use Developer Studio debugger to set breakpoints, step through an application, view the call stack, and view and alter variables.

® Debug applications, controls, and ISAPI extensions.

Chapter 5: Adding User-Interface Features

Topics

® Menus

® Toolbars

® Status bars

® Dialog bars

® Advanced techniques with menus

® Advanced techniques with toolbars

® Enabling graphics in status bar panes

® Advanced techniques for command routing

Labs

® Enabling static drop-down menus in an application

® Changing text in menus

® Enabling menu items

® Adding a shortcut menu

® Adding a time pane to the status bar

® Adding a dialog bar

® Adding a progress control to the status bar

Skills

Students will be able to:

® Add menus, accelerators, status bar menu prompts, toolbar buttons, and dialog bars to an application.

® Explain the routing of a command message.

® Dynamically change the state of a menu item.
® Describe Ownerdraw menus.
® Incorporate a context or shortcut menu into an application.
® Add additional panes and graphics to a status bar.
® Use advanced techniques with menus, toolbars, and command routing.

Chapter 6: Creating and Using Dialog Boxes

Topics

® Designing and creating dialog boxes
® Implementing the dialog box class
® Using Common Dialog Boxes
® Using List Boxes
® Using modeless dialog boxes
® Advanced dialog box handling

Labs

® Modifying resources and adding dialog boxes to an application
® Adding a modeless dialog box
® Using common dialogs
® Customizing the common dialog class
® Adding a property sheet to an application

Skills

Students will be able to:

® Define the different types of dialog boxes.
® Explain how dialog boxes are built by using MFC Library.
® Use Dialog editor to create a dialog box template.
® Use ClassWizard to create dialog box classes.
® Write code to manage dialog data exchange and data validation.
® Create an instance of the dialog box class.
® Use advanced techniques for placing data in controls of dialog boxes.
® Invoke and display modal and modeless dialog boxes.
® Extend dialog data validation.
® Customize common dialog boxes.
® Create tabbed dialog boxes and property sheets.

Chapter 7: Implementing View Classes

Topics

® Introduction to view classes
® Multiple views in single document interface (SDI) and multiple document interface (MDI) applications
® Scroll views
® Splitter windows
® CFormView
® CListView

® CTreeView

® CEditView

® CRichEditView

® Writing an application with multiple interrelated views

Labs

® Adding a Splitter bar to an application

® Adding Open File dialogs and Rich Edit view to an application

® Building a text viewer

Skills

Students will be able to:

® Describe the purpose of documents, views, templates, and frames within the document/view architecture, and how they interact.

® Describe the various types of view classes in MFC.

® Implement applications that use CScrollView, CListView, CSplitter, CTreeView, CEditView, and CRichEditView.

® Use two interrelated views in an application.

Chapter 8: Creating Database Applications

Topics

® Introduction to databases

® Recordset

® Introduction to data access with MFC

® Using Open Database Connectivity (ODBC)

® Using Data Access Objects (DAO)

® DAO versus ODBC

® Creating a DAO database application using AppWizard

® Changing the fields in a recordset

® Viewing recordsets in a forms-based interface

® Moving within a recordset

® Overriding CDaoRecordView::OnMove

® Transactions

® Editing existing database records

® Adding and deleting records

Labs

® Building a database viewer with DAO

® Building a database editor with DAO

Skills

Students will be able to:

® Describe database concepts and terminology.

® List methods of data access available in MFC.

® Explain the differences between ODBC and DAO.

® Overview of the ODBC API

® Create a database application that allows users to view, edit, add, and delete records.

Chapter 9: Creating ODBC Database Applications

Topics

® ODBC overview

® Data Sources

® Creating an ODBC database application using AppWizard

® ODBC classes in MFC

® Creating database elements

® Moving through recordsets

Lab

® Building a database editor with ODBC

Skills

Students will be able to:

® Describe the role of ODBC in applications that interact with databases.

® Identify and describe the most important issues that a simple ODBC-compliant application must address.

® Use ODBC Administrator and understand its use of the registry

® Create ODBC database applications that connect to a local database.

Chapter 10: Creating Advanced Database Applications

Topics

® Customizing a query

® Using Querydefs

® Stored queries

® Finding records within a recordset

® Accessing external data

® Relations

® Database schema creation

® Error handling, record locking, and transferring binary data

Labs

® Building an advanced database application

® Creating a new database using DAO

Skills

Students will be able to:

® Build and use queries.

® Search for specific records within a recordset.

® Access external databases by using the MFC DAO classes.

® Create databases programmatically.

® Use database relations.

Chapter 11: Building Internet Applications

Topics

® Overview of Internet application concepts and MFC support

® Microsoft Personal Web Server

® Internet Explorer object and Web Browser control

® Using the WinInet classes

® Using the WinSock classes

Labs

® Using the Web Browser control

® Using the hypertext transfer protocol (HTTP) WinInet classes

® Using the file transfer protocol (FTP) WinInet classes.

® Adding WinSock capabilities.

Skills

Students will be able to:

® Describe the Internet framework.

® Install and use Personal Web Server to administer a Web site.

® Create MFC applications that invoke Internet Explorer.

® Use the Web Browser control in MFC applications.

® Create MFC applications that use the WinInet classes to enable communication across the Internet.

® Create MFC applications that use the synchronous and asynchronous WinSock classes.

Chapter 12: Creating and Using ActiveX Controls

Topics

® Introduction to ActiveX controls

® Creating an ActiveX control with ControlWizard

® ActiveX control properties, methods, and events

® Communicating errors in ActiveX controls

® Implementing ActiveX control property pages

® Creating an enumerated property

® Data binding in an ActiveX control

® Using an ActiveX control in an MFC application

® Optimizing ActiveX controls

® Using ActiveX controls on the Internet

® Implementing dual interfaces

Labs

® Building an ActiveX control, set its properties, and add a property sheet interface

Skills

Students will be able to:

® Describe the advantages of the ActiveX control technology.

® Explain the elements of an ActiveX control.

® Explain the purpose of the Object Description Language (ODL) file.

® Explain the features of ControlWizard in creating an ActiveX control.

® Describe the primary tasks of an ActiveX control container.

® Explain the interaction between an ActiveX control container and an ActiveX control.

® Use AppWizard to create an ActiveX control container and use the Gallery to incorporate an ActiveX control into the application.

® Use ControlWizard to create skeletal code for your ActiveX control.

® Use ClassWizard to add both stock and custom properties and methods to an ActiveX control.

® Use ClassWizard to define stock and custom events that the ActiveX control will use to communicate with its container.

Chapter 13: Creating and Using ATL COM Objects

Topics

® Introduction to ATL COM Objects

® Creating an ATL COM Object

® Implementing an ATL COM Object

® Adding ATL COM Objects to a Server

® Adding a New Interface to ATL COM Objects

Labs

® Building a COM Object Server

® Building a COM Object Client Application

Skills

Students will be able to:

® Describe the purpose and benefits of ATL.

® Briefly explain the Component Object Model (COM) specification and related terminology.

® Use ATL COM AppWizard to create an in-process (.Dll) COM object.

® Use the ATL COM AppWizard to create an out-of-process (.Exe) COM object.

® Add properties and methods to ATL COM objects.

® Use a COM object in an application.

® Invoke a COM server from a client application.

® Query a COM server for other interfaces.

Chapter 14: Creating ISAPI Extensions

Topics

® Architecture of ISAPI extensions

® Support for ISAPI extensions in Visual C++

® Building an ISAPI application

® Building an ISAPI filter

® Analyzing and debugging ISAPI applications and filters

Labs

® Viewing HTTP messages

® Building an ISAPI application

® Building an ISAPI filter

Skills

Students will be able to:

® Understand the role of the ISAPI server framework.

® Create an ISAPI application using ISAPI Extension Wizard

® Create an ISAPI filter using ISAPI Extension Wizard.

Course No. 780

This course replaces course 626.

This course is intended for developers within Solution Provider and MIS organizations who will be responsible for implementing sophisticated Visual Basic solutions, and for system analysts and information managers who design Visual Basic applications, or who manage developers of those applications.

This course syllabus should be used to determine whether the course is appropriate for the student, based on current skills and technical training needs. Technical information is provided on the intended audience, course prerequisites, covered topics, lab exercises, course materials, and software.

Course content, prices, and availability are subject to change without notice.

This course updates course 626, *Mastering Visual Basic 4*, and introduces significant new content. The new material covers areas of component creation and Internet development.

At Course Completion

At the end of the course, students will be able to create applications and make executable files; write and debug code; describe how Microsoft Visual Basic accesses databases; use the Data control to work with data in a Microsoft Access database; use the data-bound grid and data-bound combo box to view and update data; use data access objects to open a database and to work with recordsets; write applications that access data using DAO and that use stored queries; create an application that accesses external data and handles multi-user locking issues, referential integrity, and rule-violation errors; declare and invoke a routine in a .dll and create an application that uses Win32 API functions; use Visual Basic to program Microsoft Excel; create code components; describe how an ActiveX control differs from an ActiveX automation server; use an ActiveX control on a Web page; create ActiveX documents and use them in Internet Explorer and Office Binder; use the Internet control to build Internet-aware applications; create a setup program and use resource files when creating applications.

Microsoft Certified Professional Exams

This course helps you prepare for the following Microsoft Certified Professional exams:

® Developing Applications with Microsoft Visual Basic 5.0

Prerequisites

This course assumes the user is an intermediate-level Visual Basic programmer. Students should be able to accomplish the following tasks:

® Create an application with multiple forms and add functionality for multiple events to the controls on those forms

® Write a function procedure and a sub procedure and invoke them from an event procedure

® Describe the purpose and use of each of the controls in the Toolbox

® Add a menu to an application

® Retrieve and validate information from a user

® Add a custom control to a project

® Describe the relationship between properties and objects

® Debug code

® Create an .exe file

The course materials, lectures, and lab exercises are in English. To benefit fully from our instruction, students need an understanding of the English language and completion of the prerequisites.

Course Materials and Software

The course workbook and lab book are yours to keep.

You will be provided with the following software for use in the classroom:

® Windows 95
® Visual Basic, version 5.0
® Internet Explorer, version 3.0
® Microsoft Excel
® Personal Web Server

Day 1

Module 1: Visual Basic Review

Topics

® Understanding Visual Basic development
® Designing the user interface
® Writing code
® Validating input and handling errors
® Compiling an application

Labs

® Creating forms
® Adding code
® Creating an error-handling routine
® Restricting input
® Create an executable program

Skills

Students will be able to:

® Create a simple application using Visual Basic and create an executable file users.
® Set properties and add code to controls on forms.
® List the files that make up a Visual Basic application.
® Validate data.
® Debug and trap errors in an application.

Module 2: Using the Data Control

Topics

® Overview of data access
® Working with the data control
® Using data control events
® Using ActiveX data-bound controls

Labs

® Creating a data entry form
® Using the DBCombo control
® Using the validate event

Skills

Students will be able to:

® Describe how Visual Basic gains access to databases.

® Differentiate between tables, fields, and records in a database.

® Describe the purpose of an index.

® Use Structured Query Language (SQL) to retrieve records from a database.

® Use the **Data** control and standard bound controls to view, add, delete, and update the contents of a Microsoft Access database.

® Describe the situations that cause the events of the **Data** control (Validate, Reposition, and Error) to occur.

® Use the data-aware **DBList** control to add data to a table.

® Describe the differences between the properties **DataField**, **ListField**, and **BoundColumn** of the data-aware **DBList** control.

® Display information in the data-aware **DBGrid** control.

Module 3: Using Data Access Objects

Topics

® Overview

® Working with **Recordsets**

® Modifying data

® Finding records

® Using queries

Labs

® Creating an navigating a Recordset

® Adding and editing records

® Finding records

® Using queries

® Disabling buttons during edit (optional)

® Using a parameter query (optional)

Skills

Students will be able to:

® Write an application that uses DAO to add, delete, update, and find data.

® Describe the differences between a table-, snapshot-, and dynaset-type **Recordset**, and determine which type of **Recordset** to use in a given scenario.

® Write an application that uses queries stored in a database.

Day 2

Module 4: Advanced Database Development

Topics

® Enforcing data integrity

® Multiple-user issues

® Accessing external databases

® Accessing ODBC databases

® Performance issues

Labs

® Handling referential integrity violations

® Multi-user issues

® Using ODBCDirect (optional)

Skills

Students will be able to:

® Handle errors caused by rule and referential integrity violations for a Microsoft Jet database.

® Use transactions.

® Open a **Recordset** for exclusive use.

® Resolve locking conflicts.

® Accesses a Microsoft SQL Server database by using ODBCDirect.

Module 5: Using Dynamic-Link Libraries

Topics

® Overview

® Declaring DLLs

® Calling DLLs

® Additional DLL information

Labs

® Locating the Windows folder

® Creating a topmost window

® Using callbacks

Skills

Students will be able to:

® Describe the purpose of DLLs.

® Map C language data types to Visual Basic data types.

® Describe the purpose of the **Any** keyword.

® Pass a null value to a DLL procedure

® Explain when to use the ANSI version of a function rather than the unicode version.

® Describe the differences between the **ByVal** and **ByRef** keywords

® Implement a callback function by using the **AddressOf** keyword

® Create a Microsoft Visual Basic application that:

- Causes the title bar of a window to flash
- Causes a form to remain as the topmost form in an application
- Uses a Windows time with a callback function

Module 6: Creating ActiveX Clients

Topics

® The Component Object Model

® Implementing Automation

® Characteristics of server components

® Creating a client in Visual Basic

® Receiving notifications from servers

® Creating a client that uses Microsoft Excel

Lab

® Controlling Microsoft Excel

® Creating a client application

Skills

Students will be able to:

® Explain the major constructs of COM.

® Explain the major constructs of Automation.

— Create a client application that can manipulate other applications through Automation, use multiple interfaces on an object, and receive notifications from a server.

Day 3

Module 7: Creating ActiveX Code Components

Topics

® Overview

® Creating objects in Visual Basic

® Working with ActiveX code component projects

® Testing ActiveX code components

® Using events

® Implementing an interface

® Additional features of code components

Labs

® Create a code component

® Debugging and error handling

® Adding component information and help

® Defining and using events

® Compiling and registering the component

® Creating an asynchronous method (optional)

Skills

Students will be able to:

® Use a class module to create an object within a Visual Basic project

® Create an ActiveX code component that exposes properties, events, and methods

® Create a client application that uses your code component

® Debug and test your code component

® Raise events from your code component

® Call your component asynchronously by using events

Module 8: Creating ActiveX Controls

Topics

® Overview

® Creating a control's user interface

® Testing a control

® Exposing properties, methods, and events

® Creating property pages

® Data binding

® Distributing a control

Labs

® Creating a control

® Adding properties and methods

® Saving and loading properties

® Raising an event

® Creating a property page

® Create a data-bound control (optional)

Skills

Students will be able to:

® Describe the benefits of using ActiveX controls.

® Describe how an ActiveX control differs from an ActiveX Automation server.

® Create an ActiveX control

® Test the debug an ActiveX control

® Expose properties, methods, and events of an ActiveX control.

® Enable the data-binding capabilites of an ActiveX control.

Day 4

Module 9: Using ActiveX Components on a Web Page

Topics

® Overview

® Downloading controls

® Scripting a control.

® Providing security for components

Labs

® Creating a setup package

® Scripting a control

® Add licensing (optional)

Skills

Students will be able to:

® Use the Application Setup Wizard to create an Internet download setup.

® Use an ActiveX control on a Web page.

® Create and test a package file for licensing

® Add Visual Basic Scripting Edition (VBScript) code to an HTML page.

® Use safety control options.

Module 10: Creating and Using ActiveX Documents

Topics

® Overview

® Working with ActiveX document projects

® Testing and debugging ActiveX documents

® Multiple document projects

® The user interface and user documents

Labs

® Converting forms to ActiveX documents

® Adding properties to ActiveX documents

® Extending user interface functionality

® Creating a document container (optional)

Skills

Students will be able to:

® Describe how ActiveX Documents differ from embedded objects, and when to use them

® Describe the function of an ActiveX document container

® Create an ActiveX project with one or more **UserDocument** objects

® Use the **Hyperlink** object to gain access to the Internet or an intranet

® Persist data for an ActiveX document

® Automate an ActiveX document

Day 5

Module 11: Creating Internet-Aware Applications

Topics

® Using sockets

® Enabling FTP and HTTP connections to the Internet

® Using the WebBrowser control

® Using Automation with Internet Explorer

® Using Personal Web Server

Labs

® Building the browser application

® Building the FTP download component

® Adding browsing capabilities to the component

® Coding a chat session application

Skills

Students will be able to:

® Enable communication between clients and servers across the Internet or an intranet by using the Winsock

control

® Enable HTTP and FTP connections to the Internet.

® Use the WebBrowser control to create applications that browse HTML pages .

® Use Internet Explorer as an Automation server from a Visual Basic application

Module 12: Application Setup and Optimization

Topics

® Creatng a setup program

® Optimizing an application

® Maximizing your productivity

Labs

® Using resource files

® Using the Registry

® Using the Setup Wizard

® Using the Visual Basic Code Profiler (optional)

Skills

Students will be able to:

® Create a setup program, either manually or with the Setup Wizard.

® Describe a variety of techniques for optimizing the performance of an application.

® Use Microsoft Visual SourceSafe.

® Use resource files.

® Use the **GetSetting** and **SaveSetting** statements to save application-specific information to an .ini file or to the Registry.

® Use the App object to log error, warning, and information events.

This section includes information about User Groups and Mailing Lists that may be of interest to Web site developers.

® User Groups

® Mailing Lists

® Microsoft Public NewsGroups

Click here to connect to find out about User Groups on the Mindshare page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

No matter how much — or little — you know about computers, a user group can help you get more out of Microsoft software. Computer user groups can take any form, from an informal group of 10 individuals meeting over a laptop and take-out food, to a nonprofit corporation with a membership the size of a small town, a board of directors, and a telephone directory-sized newsletter. User groups can serve whole communities, educational institutions, corporations, professions, or associations. They can center around a single software program or an operating platform. They can be exclusive to in-house information system managers or more general to, for example, home users who want to have more fun with their computers. Or, they can be so large as to encompass all of these things at once.

Whatever their size or form, user groups share in common the commitment to providing a platform for sharing experience, insight, and knowledge about computing, for the personal and professional enrichment of their members.

As a member of one or more of these mailing lists you can engage in discussions with diverse groups of developers. Representatives from Microsoft will also be available on these lists to help answer questions and keep you up to date on the latest developments.

Examples of technologies that are represented on mailing lists include Microsoft Visual Basic Scripting Edition, Microsoft Internet Explorer, the Microsoft Crypto, WinInet, and WebPost APIs, Microsoft ActiveX Controls, Microsoft ActiveX Scripting, Microsoft Internet Information Server, and many more.

Each Microsoft technology mailing list is available in either digest or non-digest form. The digest form compiles all postings to a particular mailing list, and sends the compilation out in a single piece of e-mail: each day. You select the form you wish to receive when you subscribe. Note that you can change the form in which you receive the list at any time.

To subscribe to a mailing list, send e-mail to the mailing list address and, in the message body, type

Subscribe *listname username@email.address*

listname is the name of the list to which you want to subscribe; *username* is your e-mail name; and *email.address* is your e-mail address.

To receive list mailings in digest form, type

Digest *listname username@email.address*

The list below is a sample of some Microsoft mailing lists. Note that some of the lists below provide an e-mail address you can contact for more information.

ActiveXControls@lists.msn.com

Development of ActiveX Controls and Containers (formerly OLE Controls and Containers), COM Objects for the Internet, and OLE Controls 96.

E-mail: oleidea@microsoft.com

ActiveXScript@lists.msn.com

ActiveX Scripting mailing list, covering development of OLE scripting engines, and the use of OLE scripting languages (other than VBScript).

E-mail: oleidea@microsoft.com

IE-HTML

Includes discussions on writing HTML, taking advantage of Internet Explorer, using new/advanced HTML abilities such as cascading style sheets (CCS1), the OBJECT tag, the SCRIPT tag, and so on, and using ActiveX controls. See the Internet Explorer Web site for more information on the browser.

VBScript@lists.msn.com

Microsoft Visual Basic Scripting Edition (VBScript) mailing list, covering all aspects of VBScript development.

E-mail: vbwish@microsoft.com

Click here to connect to Microsoft's NewsGroup search page:
{ewc mvimg, mvimage,!intjump.bmp}

The following are Microsoft sponsored public news groups:

® **Transaction Server Announcements**

(microsoft.public.microsofttransactionserver.announcements)

® **Transaction Server Administration & Security**

(microsoft.public.microsofttransactionserver.administration-security)

® **Transaction Server Integration**

(microsoft.public.microsofttransactionserver.integration)

® **Transaction Server Programming**

(microsoft.public.microsofttransactionserver.programming)

This section includes product descriptions and marketing and ordering information for various publishers of technical information for software developers.

® Advisor Publications

® The Cobb Group

® Fawcette Technical Publications

® Macmillan Computer Publishing

® Microsoft Developer Network (MSDN)

® Microsoft Press

® Microsoft Systems Journal

® Patricia Seybold Group

® Pinnacle Publishing, Inc.

Click here to connect to the Microsoft Developer Network (MSDN) page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

MSDN is the official source of development toolkits, operating systems, and development-related technical, strategic, and resource information from Microsoft. It is the comprehensive source of programming information and toolkits for those who write applications for the Microsoft Windows and Windows NT operating systems, or use Microsoft products for development purposes. The Microsoft Developer Network is an annual membership program. Members are kept up to date through regular deliveries of information in the Development Library CDs, a newsletter, and other information sources.

By becoming a member of the Microsoft Developer Network, developers ensure that they receive the most current technical and strategic information right from the source. To subscribe to the Microsoft Developer Network, call (800) 759-5474.

Many of the whitepapers in the Technical White Papers section were provided by MSDN.

Click here to connect to the Microsoft Press page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Microsoft Press is the independent publishing division of Microsoft Corporation — your source of inside information and unique perspectives about Microsoft products and related technologies. We've been around since 1984, and we offer a complete line of computer books — from self-paced tutorials for first-time computer users to advanced technical references for professional programmers.

Microsoft Press Interactive, our new multimedia publishing group, creates exciting CD-ROM based interactive training and reference products for business and home use. Microsoft Press International distributes Microsoft Press books worldwide in English language editions and in 29 local languages worldwide. In short, when you need to know more about Microsoft products, come to us. After all, who knows more about Microsoft?

Visit the Books section to review a list of Web site development-related books published by Microsoft Press.

® Books Available from Microsoft Press

For more information on how to order from Microsoft Press, go to the next section.

® Ordering Information

Microsoft Press books and CD-ROM titles can be purchased at bookstores, software resellers, or directly from Microsoft Press. Click here to connect to the Microsoft Press page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Microsoft *Press*

PO Box 141875
Austin, TX 78714-9745
Phone: (800)MSPRESS
Fax: (800) 826-5399
CompuServe: GO MSP
MSN: GOto MSPress

In Canada, contact:

Macmillan Canada
Attn: Microsoft Press Dept.
164 Commander Blvd.
Agincourt, Ontario
Canada MIS 3C7
Phone: 1-800-667-1115
Fax: (416) 293-0846)

For information about Microsoft Press publications outside the United States and Canada, please contact your local Microsoft subsidiary, see the international distributor list on the Microsoft Press Web site, or contact Microsoft Press by mail or fax:

Microsoft Press
International Coordinator
One Microsoft Way
Redmond, WA
98052-6399
Fax: 1 (425) 936-7329

Click here to connect to the Pinnacle Publishing Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Pinnacle Publishing is a leading publisher of technical newsletters, in-depth special reports, and CD-ROMs on a variety of software products. Our information products give you time-saving tips, practical techniques, and up-to-the-minute solutions to help you get the most from your software in the least amount of time. Each of the products is highly respected for the depth, quality, and readability of its contents.

Our newsletters and special reports are written by experts—software professionals, trainers, consultants, and developers—who know their software products inside and out. They share the techniques they've learned from extensive experience, helping you to be more productive.

> *"You can examine your first issue of any Pinnacle newsletter FREE. If you don't like it, return your invoice marked 'Cancel' and pay nothing. If you like it, simply return payment with your invoice to receive the remaining 11 months of your subscription. And if, at any point during your subscription, you decide you're not completely satisfied, we'll refund your money. That's our promise to you."* —Mickey Friedman, Publisher

Get More Information

You can get more detailed information about any Pinnacle Product via our web site at **http://www.pinpub.com.** You can also get more information through our Information Fax Service (IFS) by calling toll-free 800-788-1900. When prompted, enter the desired IFS numbers along with your fax number.

We Value Our Customers

Pinnacle Publishing is known for outstanding customer service and superior technical support. We do not believe service ends when a sale is made–we have a commitment to your future success. We do whatever we can to help you get the most from your Pinnacle products and the host products they support, and to keep you apprised of new tools and resources that will make your job even easier. To put it simply, our goal is to exceed your expectations every step of the way.

You can count on Pinnacle to provide useful technology and expert information now and in the future. We have plans underway for new software and publications, and we're beginning to distribute our products via on-line services. If we can do anything to make your work more successful, please let us know.

And remember: Pinnacle products are guaranteed! If a product or publication does not meet your expectations, we'll refund your money. There's absolutely no risk—and no hassle—when you rely on Pinnacle.

How to Contact Us

Pinnacle Publishing, Inc.
PO Box 888
18000 72nd Ave S., Suite 217
Kent, WA 98035
(425) 251-1900 voice
(425) 251-5057 fax
(425) 251-6218 BBS
Email: ppi@pinpub.com
Web site: http://www.pinpub.com
CompuServe: GO PINNACLE

Click here to connect to the Microsoft Web Builder page on The Cobb Group Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The Cobb Group is the world's leading publisher of software-specific journals. The Cobb Group's "how-to" publications provide a constant flow of fresh ideas, timesaving tips, and innovative techniques for applications, networking, development, operating systems, and entertainment packages.

The Cobb Group's Developer Team publishes newsletters covering a number of Microsoft products, including Visual J++, Active Server Pages, Visual Basic, and Office. They also offer Microsoft Web Builder, a multi-solution journal that focuses on developing Web sites using a variety of development tools.

Corporate Background

Founded in 1984, The Cobb Group successfully tapped into the rapidly growing market of PC users seeking information. In 1990, Inc. magazine named The Cobb Group one of the 500 fastest-growing private companies in the country. In 1991, The Cobb Group was sold to Ziff-Davis Publishing Company and today operates as part of ZD's Computer Support and Training Division. Based in Louisville, Kentucky, The Cobb Group continues to produce publications valued for their clarity, accuracy, and timeliness.

How To Contact Us

For subscriptions to journals, CD-ROMs, or resource disks, contact:

The Cobb Group
Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220
1-800-223-8720
Fax 502-491-8050
cobb_customer_relations@zd.com

You can also order sample issues at the Web site shown above.

The following sample articles from The Cobb Group are available in the Technical Articles section:

® Adding Database Functions with ASP

® Browser Detection with JScript

® Building an Online Store, Part I

® Building an Online Store, Part II

® Create an Online Gallery With ActiveX Controls

® Redirecting Users to the Login Page

® Server Side Scripting: A Primer

® Using JavaScript to Validate a Form

® Using Session Objects to Extend the Functionality of ASP Applications

® Using Session Objects with ADO

Click here to connect to the Fawcette Technical Publications web site.
{ewc mvimg, mvimage,!intjump.bmp}

Fawcette Technical Publications is the leading supplier of information products for professional Windows developers including: Visual Basic Programmer's Journal, the leading Windows development magazine; Microsoft Interactive Developer, the premier source of information on developing interactive applications, The Development Exchange, the most comprehensive online resource available for Windows development information and tools; VBITS conferences; the monthly VBCD CD-ROM; Companion Products directories; and the Windows Components Forums on CompuServe.

Fawcette Technical Publications
209 Hamilton Avenue
Palo Alto, CA 94301
Toll Free: 800-848-5523
International: 415-833-7100
Fax: 415-853-0230

Click here to connect to the Advisor Publications web site.
{ewc mvimg, mvimage,!intjump.bmp}

Through a variety of magazines, resource disks, CD-ROMs, and in-person conferences, Advisor Publications has been a leading provider of technical guidance to IS professionals and business users since 1983. Presenting the knowledge of recognized experts, and covering today's important database technology and techniques, Advisor's mission is to help corporations implement business solutions that really work.

Advisor Publications Inc.
5675 Ruffin Road
San Diego CA 92123 USA
800-336-6060
(619)278-5600
Fax (619)278-0300
E-mail advisor@advisor.com

Below is a brief description of magazines offered by Advisor Publications.

Access/Visual Basic Advisor

The only magazine devoted to solving business problems with the full range of Microsoft business software, including Access, Visual Basic, Office (Excel, Word, PowerPoint), BackOffice and Windows. Provides detailed guidance on developing applications in clear, straightforward language.

12 issues per year

Magazine Only		**Magazine & Resource Disk**	
US	$39.00	US (excluding CA)	$129.00
Canada	$59.00	CA	$135.98
Other Countries	$79.00	Canada	$169.00
		Other Countries	$199.00

Databased Advisor

The editorial authority since 1983 on all areas of PC database management and application development. Written by IS professionals for their colleagues, every issue provides practical techniques, real-world case studies, new product reviews, and extensive test drives, covering all leading tools and platforms.

12 issues per year

Magazine Only		**Magazine & Resource Disk**	
US	$39.00	US (excluding CA)	$129.00
Canada	$59.00	CA	$135.98
Other Countries	$79.00	Canada	$169.00
		Other Countries	$199.00

FoxPro Advisor

The only magazine for developing custom business solutions with Microsoft Visual FoxPro for Windows, and Microsoft FoxPro for MS-DOS, Macintosh and UNIX. Each issue is filled with hands-on information, including significant news, features articles by renowned experts, tips and techniques, reviews of related products, and answers to programming problems.

12 issues per year

Magazine Only	**Magazine & Resource Disk**

US	$49.00	US (excluding CA)	$139.00
Canada	$69.00	CA	$145.98
Other Countries	$89.00	Canada	$179.00
		Other Countries	$209.00

Internet & Java Advisor

The complete technical guide to achieving the business benefits of the online universe. Written by expert consultants and developers, each issue explains exactly how to design, build, deploy, and manage applications that use Internet, Intranet, Java, and Web technology to extend the business.

Magazine Only		**Magazine & Resource Disk**	
US	$49.00	US (excluding CA)	$139.00
Canada	$69.00	CA	$145.98
Other Countries	$89.00	Canada	$179.00
		Other Countries	$209.00

Microsoft Office Advisor

The “how-to” magazine for users of the world’s most popular business software. Provides independent guidance and expert advice to the millions of business users worldwide who rely on Microsoft Office and related technology to implement real-world business solutions.

Magazine Only		**Magazine & Resource Disk**	
US	$39.00	US (excluding CA)	$129.00
Canada	$59.00	CA	$135.98
Other Countries	$79.00	Canada	$169.00
		Other Countries	$199.00

This section provides pointers to a number of Web sites, which provide information useful to a Web site developer.

Note At the time this course was released, all Web sites listed in this section were valid and fully functional. However, as the World Wide Web evolves, you may find that one of the Internet jumps provided in this title is no longer valid. If this happens, try searching the Web for the site, or connect to the Mastering Series Web page on the Microsoft Web site to locate updated Uniform Resource Locators (URLs).

You can jump to one of the following Web sites by going to the topic and clicking the Internet Jump icon for the site:

® Microsoft
® Microsoft ActiveX SDK for Macintosh
® Microsoft BackOffice
® Microsoft Exchange Server
® Microsoft FrontPage 97
® Microsoft Interactive Media Technologies
® Microsoft Internet Development
® Microsoft Internet Information Server
® Microsoft Mastering Series
® Microsoft Press Releases
® Microsoft SQL Server
® Microsoft Visual Basic Home Page
® Microsoft Visual Basic Scripting Edition
® Microsoft Visual InterDev
® Microsoft Visual Tools
® Microsoft Windows Family
® Microsoft Windows NT Server

Click here to connect to the Mastering Series page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The Mastering Series web site offers you the following information about the titles:

® What's New

® Titles Available

® Frequently Asked Questions

® Where to Buy

® International Update

The site offers you a chance to submit feedback on the Series, as well as links to these other Microsoft web pages:

® Support

® Knowledge Base

® For Developers Only

Click here to connect to the Microsoft World Wide Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The Microsoft site offers details about Microsoft products ad technologies. Some of the places you can visit from this site include:

® Products

® Support

® Customer Guides

® Free Downloads

® Internet Center

® About Microsoft

® Microsoft Network

You can also perform a search for a key word or phrase to jump to a specific type of information on the Microsoft web.

Click here to connect to the Microsoft Visual Basic Script page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Microsoft Visual Basic Scripting Edition (VBScript) revolutionizes active content development for the Internet. VBScript is a high-performance scripting language designed to create active, online content on the World Wide Web, and allows developers to link and automate a wide variety of objects in Web pages, including ActiveX Controls and "applets" created using the Java language from Sun Microsystems Inc.

This section provides information and instructions for installing several Microsoft tools. All of the tools are on the *Mastering Web Site Development* CD-ROM in the Media\Tools directory. Each tool can be automatically downloaded from the CD by clicking the Launch Executable icon in the following sections:

Microsoft Press, the book publishing division of Microsoft, offers a wide variety of training and reference materials for developers. The following list of Microsoft Press books are excellent resources for a Web site developer.

® **ActiveX Controls Inside Out, Second Edition**

Adam Denning

An in-depth guide for C++ and Visual Basic programmers who want to build powerful custom controls and "componentware" using Microsoft's new tools and revolutionary COM (Component Object Model) technology.

ISBN: 1-57231-350-1

Price: $39.95

® **ActiveX Developer's Guide**

Mike Oliver

Here's the hands-on training that developers need for creating interactive, dynamic Web sites with Microsoft ActiveX technology. This is the book that brings together Microsoft's Web-related programming technologies — Internet Server Application Programming Interface (ISAPI), ActiveX Controls, Visual Basic Scripting, Document Objects, Internet Explorer Objects, Code Signing, and Component Download. And it takes full advantage of new Microsoft Foundation Class (MFC) optimization for ActiveX.

ISBN: 1-57231-413-3

Price: $44.99

® **Build Your Own Web Site**

Louis Kahn and Laura Logan

Establish an awesome Internet presence using Microsoft Windows NT Server and Microsoft Internet Information Server with *Build Your Own Web Site*. You'll learn how to use the most effective hardware, connections, security, bandwidth, and routing; how to set your system up; how to maintain security; and much more. A CD includes content templates, tools, and other information.

ISBN: 1-57231-304-8

Price: $29.95

® **HTML in Action**

Bruce Morris

A powerhouse collection of the techniques and tricks needed for readers to challenge the World Wide Web's most innovative sites with ultra-cool ones of their own.

ISBN: 1-55615-948-X

Price: $29.95

® **Inside Microsoft Internet Studio**

Ken Miller, Kenneth L. Spencer, and Eric L. Vincent

The definitive guide for developers.

Microsoft Internet Studio is the powerful, integrated design tool for developers building advanced sites for corporate intranets and the Internet. With this book, developers get their first close look at Microsoft Internet Studio — from and insider's perspective. For everyone who develops for the Web, uses Visual Basic, or creates intranets for the enterprise, this book is a must.

ISBN: 1-57231-583-0

Price: $39.99

® **Introducing Microsoft FrontPage 97**

Kerry A. Lehto and W. Brett Polonsky

Microsoft FrontPage 97 offers those with no programming experience a comprehensive package to create their own attractive and appealing Web pages — and even the ability to put them on line right out of the box. *Introducing Microsoft FrontPage 97* gives readers all the information they need to use FrontPage 97 effectively to create awesome-looking Web pages.

ISBN: 1-57231-338-2

Price: $24.95

® Microsoft FrontPage 97 At a Glance

Stephen L. Nelson

Microsoft Press *At a Glance* books provide a quick, visual step-by-step reference for those who turn to software books to solve specific problems and those who want just the information they need when they need it. The low price point, highly visual pages, and straightforward approach will entice beginning to intermediate as well as casual and occasional software users.

ISBN: 1-57231-573-3

Price: $16.99

® Microsoft FrontPage 97 Step by Step

Catapult, Inc.

Microsoft FrontPage offers those with no programming experience an easy way to create appealing Web pages — and get them quickly running on the Web. This book-and-CD package is the perfect companion, providing a procedural, personal training system for those who want to get up and running on Microsoft FrontPage as easily and quickly as possible.

ISBN: 1-47231-336-6

Price: $29.99

® Microsoft Internet Explorer 4.0 Step by Step

Catapult, Inc.

Microsoft Internet Explorer 4.0 Step by Step is the easiest way for people to learn the powerful new features of Microsoft Internet Explorer 4.0. Practice files and a free copy of Microsoft Internet Explorer 4.0 are included, providing users with the tools and training they need to surf the World Wide Web.

ISBN: 1-57231-514-8

Price: $24.99

® Microsoft Office 97/Visual Basic Programmer's Guide

Microsoft Corporation

Quickly develop powerful, flexible custom applications with Microsoft Office 97.

Microsoft Office 97 is the newest version of the world’s most popular office suite — and a powerful development environment. This book shows everyone — from professional developers to power users — how to make the most of it.

ISBN: 1-57231-340-4

Price: $34.99

® Official Microsoft Internet Explorer 4.0 Book

Bryan Pfaffenberger

The *Official Microsoft Internet Explorer 4.0 Book* is a widely appealing guide to Microsoft's premier Internet, intranet, and Windows 95 system browser, Internet Explorer, its many new features and add-ons, as well as a great Internet road map for all Windows platforms and the Macintosh.

ISBN: 1-57231-576-8

Price: $24.99

® Running Microsoft FrontPage 97

Jim Buyens

Here are the in-depth information and inside tips that newcomers and veteran Webmasters need for quickly designing and implementing Web pages filled with vivid graphics and special effects. *Running Microsoft FrontPage 97* is comprehensive, easy to use, well-organized, example-filled, and a great supplement to the standard documentation. In short, it's the one-volume handbook that Web users keep next to their computers and use every day.

ISBN: 1-57231-426-5

Price: $34.99

® Running Microsoft Internet Information Server 3.0

Leon Braginski and Matthew Powell

Running Microsoft Internet Information Server 3.0 takes the reader from an overview of how the Internet works with local area networks and intranets to advanced server concepts. This comprehensive book focuses on practical use and organization of Web sites and their integration into the reader's business/organization.

ISBN: 1-57231-585-7

Price: $39.99

® The Official Microsoft ActiveX Web Site Toolkit

Alan Simpson

This is the official Microsoft Web site builder for Webmasters and power users.

The Official Microsoft ActiveX Web Site Toolkit is a timely book on creating powerful interactive Web sites based on the ActiveX platform and using the latest generation of tools — Microsoft Internet Explorer 4.0 and FrontPage 97. It focuses on the largest portion of the rapidly growing Web site creation audience — Internet power users and corporate Web site developers.

ISBN: 1-57231-572-5

Price: $39.99

® Understanding ActiveX and OLE

David Chappell

This book shows you the strategic significance of the Component Object Model (COM) as the foundation for Microsoft's object technology — and clarifies the evolution of OLE technology. It also introduces you to ActiveX, the powerful new technology for the Internet.

ISBN: 1-57231-216-5

Price: $17.95

ActiveX server components enable you to package and reuse common functions, such as accessing a database and writing information to text files on a Web server.

You can access the components installed on a Web server by using an .asp file with the **CreateObject** method of the **Server** object.

Base Components

To help you create Web applications, Internet Information Server (IIS) 3.0 provides five ActiveX server components, which are referred to as base components.

The following table lists and describes the five base components provided by IIS.

Base component	Description
Advertisement Rotator	Automatically rotates advertisements displayed on a Web page, according to a specified schedule.
Browser Capabilities	Determines the capabilities, type, and version of a user's browser.
Database Access	Uses ActiveX Data Objects (ADO) to access information stored in a database or other tabular data structures.
Content Linking	Creates a table of contents for Web pages, and links them sequentially, like pages in a book.
File Access	Uses the **FileSystemObject** object to retrieve and modify information stored in a text file on the server.

For information about using the Database Access component, see Chapter 7: Creating Database-Aware Web Pages.

For information about using the Advertisement Rotator and Content Linking components, search for these terms in Visual InterDev InfoView.

Microsoft Visual Basic offers two scenarios for testing and debugging components: one for in-process components and one for out-of-process components. In the first scenario, you test a component within a single instance of Visual Basic. In the second scenario, you must run two copies of the development environment.

This section describes how to test an ActiveX DLL and an ActiveX EXE, and explains the advantages of setting a reference to a type library. It also explains how to raise errors to a client application to provide the client with error information.

This section includes the following topics:

```
; Netscape Browsers
[Mozilla/2.0 (Win95; I)]
browser=Netscape
platform=Win95
version=2.0
majorver=#2
minorver=#0
frames=TRUE
tables=TRUE
cookies=TRUE
backgroundsounds=FALSE
vbscript=FALSE
javascript=TRUE

; Microsoft Browsers
[Mozilla/2.0 (compatible; MSIE 3.0a; Windows 95)]
parent=Mozilla/2.0 (Win95; I)
version=3.0a
majorver=#3
minorver=#0
backgroundsounds=TRUE
vbscript=TRUE
javascript=FALSE

; Default Browser
[Default Browser Capability Settings]
browser=Default
frames=FALSE
tables=TRUE
cookies=FALSE
backgroundsounds=FALSE
vbscript=FALSE
javascript=FALSE
```

Microsoft Transaction Server manages components during transaction processing.

In the following illustration, three business objects work together to transfer money from one account to another. The **Debit** object debits an account and the **Credit** object credits an account. The **Transfer** object calls the **Debit** and **Credit** objects to transfer the money.

{ewc MVIMG, MVIMAGE,!W09g010.bmp}

The ACID Test

A transaction changes a set of data from one state to another. For a transaction not to fail, it must have the following properties, commonly known as the ACID (Atomicity, Consistency, Isolation, Durability) test.

® Atomicity

Atomicity ensures that all of the updates from objects in a transaction have completed their tasks and can commit. Otherwise, the transaction will abort and be rolled back to their previous state.

Atomicity is an all or nothing property. For example, in the transaction described earlier in this topic, both databases must be changed successfully or the entire transaction will fail.

® Consistency

Consistency ensures that a transaction correctly transforms the system state. To transform a system state, the transaction process temporarily makes the system inconsistent, and then returns the system to its consistent state.

The inconsistent state can occur only between two points. The first point is at the beginning of a transaction; the second point is when the transaction is ready to commit. In the transaction described earlier, money can be debited from one account and not yet credited to the other account during the transfer process. When the transaction is finished and able to commit, either both the debit and credit occurs, or neither occurs.

® Isolation

Isolation ensures that concurrent transactions are not aware of each other's partial and uncommitted results. Otherwise, they might create inconsistencies in the application state.

For example, in the transaction of transferring money, if two transfers occur at the same time, neither will know of the partial debit or credit from an incomplete transfer.

® Durability

Durability ensures that committed updates to managed resources (such as database records) survive communication, process, and server system failures. Transactional logging enables you to recover the durable state after failures.

All of these properties ensure that a transaction does not create problematic changes to data between the time the transaction begins and the time the transaction must commit.

This period of a transaction constitutes one part of a two-phase commit process. The second phase occurs when all the business objects used in a transaction process are successful and can proceed with the commitment.

Two-Phase Commit

When transactions are processed on more than one server, a two-phase commit ensures that the transactions are processed and completed on all of the servers or on none of the servers.

There are two phases to this process: prepare and commit. You can use the analogy of a business contract to illustrate the two-phase commit process.

In the prepare phase, each party involved in the contract commits by reading and agreeing to sign the contract. In the commit phase, each party signs the contract.

The contract is not official until both parties have made a commitment. If one party does not commit, the contract is invalid.

Microsoft Transaction Server coordinates and supports the two-phase commit process, making sure that all objects of the transaction can commit and that the transaction commits correctly.

Course No. 792

This course is intended for individuals who have programming experience in non-object oriented languages, such as Microsoft Visual Basic or COBOL, who want to learn how to create components for the Internet. It is also intended for Visual C++ programmers who want to learn the differences between C/C++ and Java and how to extend their existing knowledge of COM and ActiveX technologies to include Java.

This course syllabus should be used to determine whether the course is appropriate for the student, based on current skills and technical training needs. Technical information is provided on the intended audience, course prerequisites, covered topics, lab exercises, course materials, and software.

Course content, prices, and availability are subject to change without notice.

This course is intended to help developers learn how to use Microsoft Visual J++ to create Java applications.

This course can be taught in its entirety as a five-day class, or can be split into a two-day introductory and a three-day advanced course.

At Course Completion

At the end of the course, students will be able to:

® **Introductory portion (Chapters 1-6):** explain the architecture of a Java application and Java applet, and how these executables are loaded, verified, and run inside the Java Virtual Machine (JVM); list and describe the standard Java packages; use the Visual J++ development environment to create, build, and debug Java projects; build Java stand-alone console and windowed applications; build Java applets and integrate them into an HTML page; control Java applets from within an HTML page using the Visual Basic Scripting Edition or JavaScript scripting language.

® **Advanced portion (Chapters 7-14):** write Java applets that can communicate with Win32 COM objects; from a Java applet, connect to a local data source using remote data objects (RDO) or active data objects (ADO); from a Java applet, connect to a data source on an Internet server using ADO and Active Server Pages; create a COM object with Java; call a COM object on the server using Active Server Pages; compare and contrast the security model built into the Java language to the code-signing model; and sign and register a Java applet and an ActiveX control.

Microsoft Certified Professional Exams

This course helps you prepare for the following Microsoft Certified Professional exams:

® To be determined

Prerequisites

All course portions assume basic knowledge of the Internet and basic Windows user-level competency. They also assume prior programming experience in either C, Visual C++, Visual Basic, or other procedural language. While experience with COM and OLE is beneficial, it is not a requirement.

Potential students should be able to accomplish the following tasks before taking this training:

® Use a browser to browse the Web

® Demonstrate basic knowledge of HTML tags by creating a simple web page

® Use a modern development environment to create a stand-alone programs. For example:

- Use the Visual Basic environment to add controls to a form, place code in the appropriate events, and create an executable
- Use the Visual C++ environment and wizards to create a command line or Windows SDI application
- Use a third party product, such as Borland Delphi, to create an application

In addition, students attending the advanced portion should have completed the introductory portion or have equivalent experience with Microsoft Visual J++:

The course materials, lectures, and lab exercises are in English. To benefit fully from our instruction, students

need an understanding of the English language and completion of the prerequisites.

Course Materials and Software

The course workbook and lab book are yours to keep.

You will be provided with the following software for use in the classroom:

® Microsoft Visual J++ version 1.1

® Internet Information Server 3.0 for Windows NT Server, or Peer Web Services for Windows NT Workstation or Windows 95

Day 1 (Full Course)
Day 1 (Introductory Portion)

Chapter 1: Introduction to Java

Topics

® Introduction to Java

® Introduction to the World Wide Web

® Architecture of the Java language

® Introduction to the Java security model

® Choosing a tool for Web component development

® Java Web sites

Lab

® Using a Java applet

Skills

Students will be able to:

® Describe the role that Java plays in developing applications for the Web.

® Explain the difference between a Java applet and a Java application.

® Explain the purpose of the Java Virtual Machine and Just In Time compiler.

® Add an existing Java applet to a web page and set initial properties using HTML tags.

® Explain the Java security model for applets and the effect of COM objects in the security model.

® Determine when Java is the best development language for web component development by comparing and contrasting the web development capabilities of Visual J++, Visual Basic, and Visual C++.

Chapter 2: Using Microsoft Visual J++

Topics

® The Developer Studio Environment

® Creating, building, and running Java programs

® Resources and the Resource Wizard

® Debugging Java code

Lab

® Creating a basic Java program

Skills

Students will be able to:

® Use the Developer Studio Environment to manage project workspace elements and view online documentation.

® Create a simple Java application and applet using the Applet Wizard.

® Build and run a Java application and applet.

® Explain the purpose of resources and how they are translated to Java code using the Resource Wizard.

® Use the integrated debugger in the Developer Studio Environment to set breakpoints and watch expressions, and to step through code.

Chapter 3: Java Language Fundamentals

Topics

® Basic Java statements

® Creating program units

® Using objects

® Object-oriented programming in Java

® The Java API: Java Standard Packages

Labs

® Creating a simple Java program

® Creating a method

® Using objects

® Creating an object

® Subclassing an object

Skills

Students will be able to:

® Create a simple Java program that uses basic syntax elements such as variables, data types, operators, conditional statements, and looping structures.

® Explain the purpose and implementation of Java methods.

® Create, reference, and manipulate objects from within a Java program.

® Explain Java's implementation of basic object-oriented concepts, such as classes and inheritance.

® Describe the purpose of a package in Java.

® Import a package into a Java program.

Day 2 (Full Course)
Day 2 (Introductory Portion)

Chapter 4: Adding User Interface Controls

Topics

® Creating a user-interface with Java

® Adding and using basic Java controls

® Java event model

® Programming controls

Labs

® Adding controls to a Java program

® Programming controls

Skills

Students will be able to:

® Explain the relationship between the Abstract Windowing Toolkit (AWT) and the Dialog and Menu Editors in Developer Studio.

® Build a user interface for a Java program using basic controls, such as labels, buttons, text, and lists.

® Explain the Java event-handling model.

® List the events supported by the Java event-handling model.

® Add an event handler to a control that runs when the user interacts with the program interface.

Chapter 5: Adding Graphics and Media

Topics

® Java's graphics architecture

® Drawing 2-D graphics in Java

® Setting fonts in Java

® Playing video in Java

® Playing sound in Java

Lab

® Using graphics and media

Skills

Students will be able to:

® List the classes that support graphics in the Abstract Windowing Toolkit (AWT) package.

® Describe the purpose of the Dimension, Canvas, and Graphics classes contained within the AWT.

® Explain the difference between drawing and painting in a Java program.

® Write a Java program that draws simple shapes within the main program container.

® Change the font displayed by a Java program.

® Create a Java program that displays AVI files and plays WAV files.

Chapter 6: Controlling Java Applets Through Scripting

Topics

® Introduction to client-side scripting

® Using the JavaScript language

® VBScript language syntax

® Microsoft Internet Explorer object model for scripting

Lab

® Controlling Java applets through scripting

Skills

Students will be able to:

® Explain the role of scripting in building an active web page.

® Create a Java applet that exposes itself to scripting.

® Use the ID parameter of the OBJECT tag to identify an object for scripting.

® Explain the differences between VBScript and JavaScript.

® Use the SCRIPT tag to add scripting to a web page.

® Describe how JavaScript differs from the Java language.

® Explain how VBScript differs from Visual Basic for Applications.

® Employ the Internet Explorer Object Model for Scripting to manipulate web page elements from script code.

Day 3 (Full Course)
Day 1 (Advanced Portion)

Chapter 7: Adding Frames, Dialog Boxes, and Menus

Topics

® Java Window architecture

® Creating dialog boxes with the Resource Editor

® Using Layout Managers

® Adding menus and menu items

Lab

® Adding frames, dialog boxes, and menus

Skills

Students will be able to:

® Explain the relationship between frames, windows, and dialogs in Java.

® Use the Java Layout Manager to place controls within a window.

® Create a dialog box using the Dialog Editor in the Developer Studio Environment and convert it to a Java resource.

® Define the purpose of the Windows Dialog Layout Manager.

® Add a menu, with menu items, to a window.

Chapter 8: Exception Handling

Topics

® Introduction to error-handling and exceptions

® Java exception handling

® Using exception handling

Lab

® Exception handling

Skills

Students will be able to:

® Describe the Java exception-handling architecture.

® List some common uses for exception handling routines in a Java program.

® Create an exception-handling routine in Java code using the **try**, **catch**, and **finally** statements.

® Create a user-defined extension to the exception hierarchy and implement this new exception into a program using the **throw** statement.

Chapter 9: File I/O and Network Capabilities

Topics

® Introduction to File I/O and streams

® Working with simple streams

® Inter-Process and thread communications

Labs

® Working with files

® Using network capabilities

Skills

Students will be able to:

® Explain the purpose of streams in implementing file input and output in a Java program.

® Create a Java program that opens a file and reads from it, and then writes to another file on the user's hard drive.

® Explain basic networking concepts and how Java fits into a network model.

® Describe the networking features supported by Java and the limitations of these features.

® Write a Java program that retrieves files from an Internet server.

® Define the term "socket" and describe Java's support of sockets.

® Write a Java program that can utilize information passed by a URL.

Day 4 (Full Course)
Day 2 (Advanced Portion)

Chapter 10: Creating and Using Threads

Topics

® Introduction to threads and concurrent programming

® Using threads in a Java program

® Advanced threading techniques

® Thread usage

® Debugging multi-threaded programs

Lab

® Use threads to download a file containing the course description in the background while application performs another task

Skills

Students will be able to:

® Define "thread". Explain why multi-threaded applications are necessary.

® Use the Applet Wizard to create a multi-threaded application or applet.

® Explain the role of the **Runnable** interface in implementing a multi-threaded application.

® Use the **start**, **stop**, **sleep**, **suspend**, and **resume** methods of the **Thread** object to control how and when a thread runs.

® Control thread execution using synchronization and locking.

® Use threads to create animation in a Java program.

® Describe how threads are used in the Java event model.

Chapter 11: Accessing COM Objects from Java

Topics

® Introduction to COM and OLE

® Java and COM

® Accessing a COM object from within Java

® Java Applets and COM objects

Lab

® Using COM objects

Skills

Students will be able to:

® Define the Component Object Model (COM).

® List the Standard OLE Services.

® Explain the security implications of adding COM objects to a Java program.

® Write a Java program that creates an instance of an application as an Automation server on the client.

® Setup a remote Automation component through dcomcnfg.exe.

Chapter 12: Database Access from Java

Topics

® Introduction to Microsoft database technologies

® Introduction to Remote Data Objects (RDO)

® Using RDO from a Java application

® Introduction to Active Data Objects (ADO)

® Using ADO from a Java application

Lab

® From Lab 9, replace the server-side applet that reads from a file so that it uses RDO or ADO to create a Java application that retrieves course information from a server database

Skills

Students will be able to:

® Describe the role of ODBC, RDO, and ADO in database applications.

® Explain how ADO and RDO are used to implement Java database applications over the Internet.

® List the most commonly used objects in the RDO and ADO object hierarchies.

® Write a Java application that adds, deletes, updates, and finds data using RDO or ADO.

Day 5 (Full Course)
Day 3 (Advanced Portion)

Chapter 13: Creating COM Objects with Java

Topics

® Creating COM objects with Java

® Server-side scripting using Java components

Lab

® Creating a COM object

Skills

Students will be able to:

® Develop a COM object using Java.

® Register a Java class as a COM class using JavaReg.

® Compare and contrast server-side and client-side processing and determine the most appropriate uses for each.

® Call a Java component on the Internet server using server-side scripting.

® Call a Java component on the Internet server using the OBJECT tag.

Chapter 14: Security and Distribution

Topics

® Microsoft security framework and Java

® Creating, signing, and using Java executables

® Licensing components

Lab

® Distributing Java programs

Skills

Students will be able to:

® Explain the difference between "trusted" and "untrusted" Java programs.

® Define code signing and Authenticode and how these technologies relate to Java programs and interact with the web browser

® Build a CAB file for use in distributing a Java program over the Internet.

Microsoft Transaction Server (MTS) supports both objects that need transactions and objects that don't. Regardless of whether an object uses transactions or not, every object runs in an activity. An activity begins when a client calls an object method. The object may create many other objects in the process, but the activity ends when the original call returns.

If an object requires a transaction, Microsoft Transaction Server creates the transaction when the object is called. When the object returns to the client, the transaction either commits or aborts. When you place components in Microsoft Transaction Server, all of the infrastructure for processing and managing a transaction is provided for you.

To see an animation that shows how Microsoft Transaction Server processes a transaction, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

Transaction Server Components

Microsoft Transaction Server manages transaction server components, which are ActiveX server components.

You create transaction server components specifically for transactions. These components are capable of notifying MTS of their work status, which is either complete or aborted.

Components can be located on the same computer as Microsoft Transaction Server or they can be distributed across multiple computers. If they reside on different computers, they are still created and accessed by Microsoft Transaction Server.

For more information about ActiveX server components, see Chapter 8: Creating ActiveX Server Components.

Activities

All objects run in activities. An activity is a set of objects that run on behalf of a base client application. An activity begins when a client calls a method of an object. The activity ends when the original call to the method is returned back to the client.

When an object runs in an activity, it can create additional objects to perform work. All of these objects will run within the same activity and can be viewed as running on a single logical thread. In other words, from the time the client calls an object on MTS until that call returns, all objects created as a result of that call can be thought of as running on one thread. This simplifies writing components for Microsoft Transaction Server.

Context Objects

MTS creates a **Context** object for each transaction server component. As the component runs within the activity, this **Context** object tracks the work done by the component as well as its security information. This frees the object from tracking its own work.

When multiple objects participate in a single transaction, the associated **Context** objects work together to track the transaction. This ensures that the transaction is consistent for all objects. The **Context** objects guarantee that the whole transaction either commits or aborts.

Microsoft Transaction Server maintains the relationship between transaction server components and their associated **Context** objects.

Thread Management

Microsoft Transaction Server manages threads for you. Transaction server components should not create threads and should never terminate a thread that calls a DLL. Microsoft Transaction Server synchronizes multiple objects that are in the same activity. This means that only one thread runs in an object at any given time. Creating additional threads in an activity can create unexpected concurrency problems.

If a multi-threaded client makes multiple calls to a transaction server component, concurrency problems may occur because the component will run simultaneously in multiple activities. To avoid concurrency problems, build apartment-threaded transaction server components and make them stateless. For information about stateless components, see Creating a Stateless Object in this chapter.

To run a transaction on Microsoft Transaction Server, you must first create a new package to hold all components of a transaction. You use a package to distribute a set of components.

A package also creates a place where the security of components can be verified. Once you have placed all components of a transaction in a package, you can make the package available to other users.

To summarize, a package is:

® A collection of components that performs related tasks for an application.

® A process that hosts components when they run in MTS.

® A trust boundary that enables you to control security for a group of components.

To see a demonstration of how to create a new package, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

u **To create a new package**

1. In the left pane of the Microsoft Transaction Server Explorer, select the computer on which you want to create a new package.
2. Double-click the Packages Installed folder.
3. On the **File** menu, click **New**.
4. In the Package Wizard, click the **Create an Empty Package** button.
5. Enter a name for the new package, and then click **Next**.
6. In the **Set Package Identity** dialog box, enter a name for the package identity and then click **Finish**.

Setting Package Properties

Once you have created a package, you can set package properties, such as how the package is accessed, how it participates in the security system, and how it ends when the system is shut down.
{ewc mvimg, mvimage,!tip.bmp}

To set properties for a package, right-click the package in the Microsoft Transaction Server Explorer, and then click **Properties**. The **Package Properties** dialog box will be displayed.

To see an illustration of the **Package Properties** dialog box, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

The following table describes the package properties and where you can set them.

Property	Tab	Description
Description	General	Displays a description of the package.
Authorization	Security	Enables Microsoft Transaction Server to check the security credentials of any client that calls the package.
Process Shutdown	Advanced	Determines whether the server process associated with a package always runs, or whether it shuts down after a specified period of time.
Account	Identity	Specifies the user who has access to the package. The default value, **Interactive User**, specifies the user currently logged onto the Windows NT server.

Once you have created a new package, you can add components to manage related business services. A component can be included only in one package on a single computer, so you must decide how to combine components into packages.

To see a demonstration of how to add a component to a package, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

u **To add a component to a package**

1. Double-click the Packages Installed folder, and then click the package in which you want to install a component.
2. Double-click the Components folder.
3. On the **File** menu, click **New**.
4. Click the **Install New Component(s**) button.
5. In the **Install Components** dialog box, click **Add Files** to select the component. The component should include all of the files that are used to implement the component.
6. In the **Select Files to Install** dialog box, select the files you want to add, and then click **Open**.
7. In the **Install Components** dialog box, click **Finish**.

{ewc mvimg, mvimage,!tip.bmp}

Setting Component Properties

When you set the properties of a component at design time, you:

® Determine the process in which the component will run.

® Define the role of the component within a transaction.

To set the properties of a component, you right-click the component in the Microsoft Transaction Server Explorer, and then click **Properties**.

Transaction Property

Each transaction server component has a transaction property. Whenever an instance of a component is created, Microsoft Transaction Server checks the transaction property of the component to determine whether or not it needs a transaction to do its work.

Most Transaction Server components are marked either **Supports Transactions** or **Requires a Transaction**.

The following illustration shows the different options for setting transaction options on the **Transaction** tab. Click each option on the illustration for information on the option.

{ewc MVIMG, MVIMAGE,!W09G035.shg}

Activation Property

You use an activation property of a transaction server component to specify the process where a component will run when activated. In general, you will want the component to run in the transaction server process to prevent any faults that may occur from crashing other processes.

When a component runs in the same process as other components in its package, it eliminates the unwanted side effect of different components running in different packages. To ensure that a component runs in the appropriate process, select **In a Server Process** on the **Activation** tab.

For changes to the activation property to take effect, you must restart the server process. To restart the server process, right-click the computer icon in the Microsoft Transaction Server Explorer, and then click **Shutdown Server Processes**.

The following table describes the property settings for a transaction server component.

Setting	Description
Transaction Server Environment	Will run in the same process as MTS.

In the Creator's Process	Will run in the same process as the component.
In a Separate Process on This Computer	Enables you to specify a different computer on the network, in which the component will run.

This specifies that the component must execute within the scope of a transaction. When a new object is created, the context of the object inherits the transaction from the context of the client. If the client does not have a transaction, Microsoft Transaction Server automatically creates a new transaction for the object.

This specifies that the component must always run within its own transaction. When a new object is created, Microsoft Transaction Server automatically creates a new transaction for the object, regardless of whether its client has a transaction or not.

This specifies that the components in a package will run within the scope of their client's transactions. When a new object is created, its context inherits the transaction from the context of the client. If the client does not have a transaction, the context will be created without one.

This specifies that the components in a package should not run within a transaction.

You can easily modify your existing business components so that they become part of a transaction as ActiveX server components.

When an object is created, Microsoft Transaction Server creates a corresponding **Context** object.

A **Context** object eliminates the need to call any transaction functions, such as **BeginTrans** or **EndTrans**. You can program an object to do its work, such as modifying data in a database, and inform the **Context** object if the work has completed successfully or if the work must be undone.

For more information about **Context** objects, see Using Microsoft Transaction Server in this chapter.

Getting a Context Object

To get a reference to a **Context** object, you call **GetObjectContext**. This function returns a reference to the **IObjectContext** interface on the **Context** object. With this reference, you can call any method of the **IObjectContext** interface.

This topic discusses three of these methods: **SetComplete**, **SetAbort**, and **CreateInstance**. For information about all **IObjectContext** interface methods, go to the Help in Microsoft Transaction Server and search for "IObjectContext."

The following Visual Basic 5.0 example code calls **GetObjectContext** to get an **ObjectContext** object.

```
Dim ctxObject As ObjectContext
Set ctxObject GetObjectContext()
```

To call the **GetObjectContext** function in Visual Basic 5.0, you must set a reference to Microsoft Transaction Server 1.0 Type Library (mtxas.dll) by clicking **References** on the **Project** menu.

SetComplete Method

Each public method of your business object should indicate whether it has completed work successfully or unsuccessfully. If the method has completed successfully, it calls the **SetComplete** method on the **ObjectContext** object before returning from the method call.

The **SetComplete** method informs the **Context** object that it can commit transaction updates and can release the state of the object along with any resources that are being held. If all other objects involved in the transaction also call **SetComplete**, the **Context** object will commit the transaction updates of all objects.

The following example code adds a new customer record to the Customers table in the database, and calls **SetComplete** to indicate it has completed work successfully.

```
Dim ctxObject As ObjectContext
Set ctxObject GetObjectContext ( )
Set conn = CreateObject("ADODB.connection")
conn.Open "DSN=dbcentral;UID=User;PWD=password;"
conn.Execute sqlstr 'sqlstr is SQL that adds the new customer
ctxObject.SetComplete
```

SetAbort Method

If the **SetComplete** method fails, it must call the **SetAbort** method of the **ObjectContext** object before exiting. **SetAbort** informs the **ObjectContext** object that the transaction updates must be rolled back to their original state. If an object involved in a transaction calls **SetAbort**, the updates will roll back even if other objects have called the **SetComplete** method.

The following example code attempts to add a student to a class. If the student is already enrolled in the maximum number of classes, he or she cannot be added to a new class, so the **SetComplete** method will fail. The example calls the **SetAbort** method to undo any work.

```
If OverEnrolled(studentid) Then
   ctxObject.SetAbort
   Exit Function
```

```
End If
```

CreateInstance Method

An object will often create and use other objects to complete a transaction. To maintain the principles of an atomic transaction, the other objects must also fall within the scope of the transaction. All objects must commit, or all must abort.

For information about the concept of an atomic transaction, see Transaction Processing Concepts in this chapter.

To create a new object for a component, you call the **CreateInstance** method of the **Context** object. **CreateInstance** creates the new object and includes it as part of the existing transaction.

Note In the Microsoft Transaction Server Explorer, if one of the transaction properties of a component is set to **Requires a new transaction**, the component will always have a new transaction.

The **CreateInstance** method takes a parameter and the progID of the object being created. It uses the following syntax:

objectcontext.**CreateInstance**

The following Visual Basic 5.0 example code creates a new **Account** object that credits a checking account with $500.00.

```
Set CheckAccount = ctxObject.CreateInstance("Checking.Account")
CheckAccount.Credit(500)
```

In this example, if the component that creates the **Account** object fails and calls the **SetAbort** method, the crediting of the checking account will roll back.

While an object is active, it maintains data. An object is referred to as "stateful" if data is maintained across multiple client calls. An object is "stateless" if the data is reset with each client call.

Stateless Objects

Microsoft Transaction Server objects should be stateless. Using stateless objects provides the following benefits:

- Helps ensure transaction isolation and database consistency by not introducing data from one transaction to another.
- Reduces the server load by not storing data indefinitely.
- Improves scalability because of the reduced server load and because there are fewer internal data dependencies in the stateless object.

Components built with Visual Basic 5.0 have an Initialize and Terminate event that you can use to create and free resources that the component needs to run. Data created in the Initialize event and maintained across multiple client calls is stateful data and should be avoided. If the component requires data to be created at startup, the component should expose the Activate and DeActivate methods instead.

For more information on the Activate and DeActivate methods, go to Help in the Microsoft Transaction Server and search for Activate and DeActivate.

Creating Efficient Objects

There are a number of ways in which you can improve the efficiency of the objects you manage using Microsoft Transaction Server.

- Pass arguments by value (**ByVal)** whenever possible. The **ByVal** keyword minimizes trips across networks.
- Use methods that accept all of the property values as arguments. Avoid exposing object properties. Each time a client accesses an object property, it makes at least one round-trip call across the network.
- Avoid passing or returning objects. Passing object references across process and network boundaries wastes time.
- Avoid creating database cursors. Cursors create a large amount of overhead. Whenever you create a **Recordset** object, ActiveX Data Objects (ADO) creates a cursor. Instead of creating **RecordSet** objects, run SQL commands whenever possible.
- When making updates keep resources locked for as short as time as possible. This will maximize the availability of resources to other objects
- Enable Microsoft Transaction Server to run simultaneous client requests through objects by making them apartment threaded. In Visual Basic 5.0, you make objects apartment threaded by selecting the **Unattended Execution** option for your project properties.

The complete title of this white paper is *Using Active Server Pages with Microsoft Internet Information Server 3.0*.

Abstract

Microsoft Internet Information Server (IIS) 3.0 introduces Active Server Pages (ASP Files), the technology formerly code-named “Denali.” Active Server Pages let organizations combine ActiveX Scripts and ActiveX Server Components to easily create dynamic content, and powerful Web-based applications.

Active Server Pages enables component-based Web application development in any language, including Java. It is easy to learn; it provides, an open development environment, and it allows developers to build truly “browser-independent” Web solutions.

The combination of Active Server Pages in IIS 3.0 running on Windows NT Server 4.0 provides the ideal platform for developing personalized content and powerful Web-based business solutions for corporate intranets and the Internet.

Introduction

Organizations are looking to Internet technology to improve productivity, reduce costs, and provide access to existing information and knowledge in new dynamic and interactive ways. Businesses want to run Web-based applications on their servers, which will allow them to realize the advantages of providing users access to “Information At Your Fingertips.” For example you can:

® Put your employee handbook online, rather than printing copies that are obsolete soon after publication. This also reduces administrative costs by allowing employees to access and update their personal information such as address and health plan benefits.

® Tie your online store to your existing inventory database and order processing system.

® Give every visitor to your site a personalized view of only the information they are interested in, and automatically flag what is new since their last visit.

While much of this can be done today through creative programming tricks, the challenge until now has been to find a technology that is easy to use, open, scaleable, and takes advantage of existing skills and investments.

Microsoft Internet Information Server (IIS) 3.0 was designed to be a powerful, open platform for developing Internet and corporate intranet applications with Active Server Pages (ASP Files). IS professionals and Webmasters can combine ActiveX scripts and ActiveX Server Components running on the server to create a new generation of server-based solutions for the Web solutions.

This White Paper is designed to provide a detailed overview of the various building blocks that Active Server Pages make available to Web developers. These topics are included:

Internet Information Server 3.0 introduces Active Server Pages, which enables HTML authors and Web developers to intermix HTML with in-line scripting using almost any authoring tool. The scripts can reference components running on the local server — or any other server — to access databases, applications, or process information. When a browser requests a ASP File, it is processed by the server, and the page is returned to the client as standard HTML.

The following topics are included in this section:

When using IIS 3.0, developers are not required to use a proprietary scripting language to create Web applications — Active Server Pages are compatible with any ActiveX Scripting language. Active Server Pages include native support for Visual Basic Scripting Language (VBScript) and Jscript. Third parties will be providing support for other languages such as REXX, Perl, and Tcl through plug-ins. Multiple scripting languages can even be used interchangeably in the same ASP File.

ActiveX Server Components can be created in virtually any language. This includes Java, Visual Basic, C++, COBOL, and more.

Active Server Pages make it easy for HTML authors to “activate” their Web pages on the server. Customized pages, and simple applications can be developed immediately. Instead of writing complicated CGI programs in languages like Perl and C to generate personalized content for each user, a Web developer can use an Active Server Page to do all the work. In the following simple example, VBScript is used to display both the current time and the type of browser the client is using.

```
<HTML>
<HEAD>
<TITLE>Sample Web Page</TITLE>
</HEAD>
<BODY>
<P>
Hello <%= Request.ServerVariables(“REMOTE_USER”) %>
The time here is <%= now %>
Your browser is <% = Request.ServerVariables(“http_user_agent”) %>
</BODY></HTML>
```

Example 1. HTML with in-line VBScript

The Web brings together teams from many disciplines — graphic artists, HTML authors, programmers, publishers, and more. The challenge is to enable them to work together efficiently, and make changes without upsetting each others work. Dynamic content today often requires elements of design, logic, and content to live together. This makes Web development difficult as changes require wading through lines of Perl or C code, and an inadvertent change could damage the program or the HTML formatting.

Through the use of scripting and components, Active Server Pages allow you to separate the programming to access data in databases and applications, from the design and other content of a Web page. This helps to ensure that developers can be free to focus on writing their business logic in components without worrying about how the output looks. Conversely, it frees designers creating HTML layouts to use familiar tools to modify the page as they see fit. Scripting is the “glue” that ties them together.

In the example below, a form is used to pass a ticker symbol request in the URL to the ASP Files. The first part of the ASP File calls a component that talks to a stock price server. Properties of this object, such as opening and closing price, can then be easily inserted in the HTML. The programmer can work in any language, and only needs to worry about how to talk to the stock price server. The HTML author only needs to know how to script the component, and does not care how the stock price server works.

{ewc mvimg, mvimage,!illust.bmp}
Example 2. Active Server Pages let you separate content and logic

To prevent the need for manual recompilation whenever a change is made, just-in-time compiling automatically recompiles the ASP Files upon the next request, and loads it into the server cache. So, when building your site, by simply saving the file and refreshing the page, changes to ASP Files can be previewed immediately in your browser.

Active Server Pages files provide a browser-neutral approach to application design. Because all of the application logic to generate dynamic content can be executed on the server, developers do not have to worry about what browser is used to view the site. Browsers “see” the results of an ASP File as a normal HTML page.

Active Server Pages provide a server-side scripting environment to create and run dynamic, interactive, high-performance Web server applications. Server-side scripting enables your Web server to perform the work involved in generating customized HTML pages. For example, you can build different views based upon who the user is, what browser they are using, where they've been on your site, or what they have purchased in the past.

Scripting languages are an intermediate stage between HTML and programming languages such as C, C++, and Visual Basic. HTML is generally used for formatting and hypertext linking purposes. Programming languages are generally used for giving complex instructions to computers. Scripting languages fall somewhere in between, much like macro languages in many desktop applications.

Active Server Pages support any ActiveX scripting language through the use of scripting "engines." Scripting engines are the Component Object Model (COM) objects that process scripts. IIS 3.0 will include native support for VBScript and JScript, and plug-ins are available for REXX, Perl, Tcl, and other scripting languages.

Active Server Pages makes it possible for the Web developer to use a variety of scripting languages. This is because scripts can be processed on the server side, as opposed to the client side. In fact, several scripting languages can be used within a single ASP File. This can be done by identifying the script language in a simple tag at the beginning of the script sequence.

For example, the following script example would indicate that the upcoming script sequences are to be processed by Active Server Pages as JScript code and Visual Basic Scripting Edition code, respectively:

```
<HTML>
<SCRIPT LANGUAGE=JScript RUNAT=Server>
<JScript code here>
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
<VBScript code here>
</HTML>
```

Example 3. Using JScript and VBScript on the same ASP page

A compelling advantage of IIS 3.0 is that it will enable component-based development of Web solutions with support for ActiveX Server Components. Active Server Pages allow you to run ActiveX Server Components developed in any language, such as C++, Visual Basic, Java, or COBOL. While Internet Information Server will continue to support CGI and ISAPI programs for Web-specific applications and filters, Active Server Components offer a powerful, component-based approach for applications development.

This section contains the following topics:

ActiveX Server Components, formerly known as OLE Automation Servers, are designed to run on your Web server as part of a Web application. These components allow you to extend the functionality of your script behind the scenes — no interface is involved in running them.

ActiveX Server Components are built on a popular standard. This ensures that most programmers are already familiar with developing components and that existing development tools can be used to create these components. Moreover, there are large numbers of ActiveX Server Components already available, and ready to be used as building blocks for Web-based applications.

Depending upon how they are written, ActiveX Server Components can also be run on a Web browser and used in other environments outside the Web server, such as traditional client-server applications or application plug-ins.

Active Server Pages includes a number of “built-in” server and application building objects. These objects free developers from the “grunt-work” of writing code to access details about incoming requests from clients, managing the application state, handling cookies, and assembling the response. These intrinsic objects include the following:

- **Request and Response.** The *request* object provides access to any information passed into the script with the HTTP request. This includes information from cookies, forms, URL queries, and HTTP headers. The *response* object is used to build the response, including setting cookies, page expiration, and full control of the HTTP output stream.
- **Application and Session.** These objects are designed to make state management easier — managing a state across a number of users and applications has typically been difficult in Web-based solutions. The *session* object is used to store information needed for a particular user-session. Variables stored in the Session object are not discarded when the user jumps between pages in the application; instead, these variables persist across the entire site. The server destroys the Session object when the session expires or is abandoned. The *application* object allows properties to be set that share information among all users of a given application. There are Lock and Unlock methods to ensure that multiple users do not try to alter a property simultaneously.
- **Server.** The *server* object allows scripts to create instances of ActiveX Server Components, and thus extend the Active Server Pages environment with new capabilities. The server object provides access to methods and properties on the server. Most of these methods and properties serve as utility functions. Without the server object, it would not be possible to access components from your Web application.

To help you create Web applications, Internet Information Server 3.0 also provides several base components.

The following topics are included in this section:

® ActiveX Data Objects Component
® Content Linking Component
® Filesystem Component
® Browser Capabilities Component
® Advertisement Rotator Component

The components outlined above are just the beginning. ActiveX Server Components provide excellent opportunities for third parties as organizations move to deploy Web-based solutions across intranets and the Internet. There is a limitless number components needing to be created. Many of these will be of interest to all Web developers, while others will be designed to create custom business applications. There are currently more that 30 independent software developers writing new ActiveX Server Components for IIS 3.0, with many more on the way.

Web developers can also create components themselves using, for example, the new Microsoft Visual Basic Control Creation Edition or Microsoft Visual J++ development software.

IIS 3.0 provides a platform to not only create dynamic, personalized Web sites today, but also provides an infrastructure for powerful Web solutions in the future. This is made possible by Internet Information Server 3.0 which combines the Win32 API, open Internet standards, and the open ActiveX standard. Together they provide the best platform for Web application development.

This section contains the following topics:

IIS 3.0 brings together the key technologies needed to run business applications on either the Internet or on corporate intranets. For example, while there are many catalogs online, very few support online transactions due to the difficulty of linking to inventory and order processing systems. By allowing organizations to move towards a component-based approach to application development, Active Server Pages help to ensure that powerful solutions can be built both quickly and easily.

IIS 3.0 enables truly component-based Web development through the support of the ActiveX Server Components described above. IS professionals can purchase components from ActiveX Server ISVs, or reuse existing components to build powerful intranet solutions. Active Server Pages use these components to access information, and publish it to the Web through scripting and HTML markup. This approach to application development offers numerous advantages:

- **Rapid Application Development.** Using a component-based approach to building applications is much faster than building a complex CGI program from the ground-up.
- **Browser Independence.** Applications can easily produce a “layout” that takes advantage of the browser’s capabilities.
- **Reuse**. Based on standard interfaces, ActiveX Server Components can be used by any OLE Automation Server. This helps to ensure that these components can be re-used outside of the Web paradigm. For example, an ActiveX Server Component that provides stock-ticker functionality can be used in either Microsoft Internet Information Server, or Microsoft Excel.

Because Active Server Pages uses the same scripting and component model as Microsoft Internet Explorer, developers can choose to run scripts and components on the server and/or the client. This allows the most efficient use of network bandwidth and server capacity for a given application. For example, a script on the client can check the contents of a form for missing data before passing it on to the server for processing.

Running components on the server, also places the business logic closer to the data, which has advantages in terms of efficiency and security. Web browsers make it very easy to take components and view client-side script code. With Active Server Pages, stealing scripts or components is not possible. The source code lives only on the server and is executed to generate basic HTML.

The components that make up an ASP-based application are usually run in the same process as IIS 3.0 for the greatest efficiency. They communicate with each other and with the server using the Component Object Model (COM).

As developers build richer applications, they will want to run components across several servers and on clients. This is enabled by Distributed COM (DCOM). DCOM takes care of all the remote procedure call (RPC) magic to make it work, so developers can use the same code no matter where the object is running. Components can talk to each other in a consistent way, using the same interfaces.

As organizations move towards deploying component-based solutions on the Web, it is critical to provide a path to easily and cost-effectively “scale-up” these applications with full transaction support. Microsoft will soon release Microsoft Transaction Server, code-named “Viper,” a product that integrates component-based applications with transaction support.

The Transaction Server automatically provides applications with transaction support, so that companies can rapidly build and easily modify server applications without sacrificing mission-critical reliability and scalability. Viper is designed to work with Internet and industry standards — including HTTP, DCOM, and databases that support X/Open’s XA transactional protocol — so that businesses can preserve investments in existing mainframe and UNIX systems while deploying modern applications using component software.

Microsoft Transaction Server combines the best features of transaction processing monitors, reliability, and scalability, with the best features from object request brokers, distributed services, and components. Transaction Server provides the vital application infrastructure developers need and do not want to develop themselves. ISVs that develop server solutions estimate that building this plumbing into their products consumes 30-40% of their development costs. Active Server Page applications will be able to plug right into the Transaction Server. Examples of Transaction Server services include:

® Managing low-level operating system processes and thread pools

® Building and managing server processes

® Registering servers with the directory

® Synchronizing access to shared data and resources across multiple client requests

® Distributed security

® Management and configuration

As organizations begin to evaluate tools and products that enable component-based Web development, it is important to find a vendor that offers full transaction support. The combination of Internet Information Server 3.0, with Active Server Pages, and the Microsoft Transaction Server is designed to deliver on the need for Web-based solutions that can scale to Enterprise-level applications.

Microsoft Internet Information Server 3.0 with Active Server Pages provide the ideal platform for creating and managing dynamic server-side applications that can be deployed over the Internet and corporate intranets.

Internet Information Server 3.0 allows Web site developers to manage content, design, and application logic as separate components, so team members can focus on their specialty. Virtually any scripting and programming language can be used, while still supporting any browser. It does all of this within a no-compile development environment that frees project teams from the expense and delays of having to recompile every time an element is changed.

For organizations that are interested in reusing existing investments, purchasing turnkey solutions, or building new applications from the ground up, IIS 3.0 is the ideal environment. It simplifies the development of Web-based solutions, takes advantage of investments in languages, OLE, and existing applications, while providing a robust framework for the future.

For More Information

For the latest information on Windows NT Server, check out the Windows NT Server Forum on the Microsoft Network (GO WORD: MSNTS), or click the icon below to visit the NT ServerWeb site.
{ewc mvimg, mvimage,!intjump.bmp}

ActiveX Data Objects (ADO) provide high-performance connectivity to any ODBC-compliant database or OLE DB data source. ActiveX Data Objects allows Web developers to easily link a database to an “active” Web page to access and manipulate data. This enables putting a Web “front-end” on a legacy system or developing a new database-driven application for the Web. Unlike the Internet Database Connector (IDC), the ADO component can be “driven” using any ActiveX scripting language from a single ASP page.

ASP Files using ActiveX Data Objects can dynamically request the information from a database. For example, changes in an inventory or pricing database can be immediately reflected for every user — without touching HTML code.

The Content Linking Component manages a list of URLs so that you can treat the pages in your Web site like the pages in a book. You can use the functionality of the Content Linking Component to automatically generate and update tables of contents, and navigational links to previous and following Web pages. This is ideal for applications such as online newspapers and forum message listings.

The Content Linking Component references a Content Linking list file that contains the list of the linked Web pages. This list is stored on the Web server. A “stream” of pages can be managed and rearranged dynamically without worrying about broken links. Adding, deleting or moving pages requires changing the order of the pages in the list file. It is unnecessary to edit the HTML on the individual pages to provide navigation.

This component provides access to reading in text files stored on the server. By providing filesystem access, developers do not need to write their own code to open and close files on the file system, as most scripting languages are not allowed direct file access.

Using the Browser Capabilities Component, ASP Files can recognize the capabilities of a requesting browser, and dynamically optimize the layout and content. This ensures that the Webmaster does not have to create a series of duplicate pages for each browser. For example, the microsoft.com site has one view for ActiveX enabled browsers, a second for frames-enabled browsers, and a third for browsers that don't support frames. ASP Files handle the customization, so there is only one copy of the content.

In the following example, the Browser Capabilities Component is being used to determine how to deliver dynamic advertising. If the client's browser supports ActiveX, a client-side control is sent. Otherwise, the advertising component is run on the server, which sends only a graphic image to the client.

```
<%
  Set OBJbrowser = Server.CreateObject("MSWC.BrowserType")
  If OBJbrowser.ActiveXControls = TRUE Then
%>
  <OBJECT
CODEBASE="/AdvWorks/Controls/nboard.cab#version=5,0,0,5"
WIDTH=460
HEIGHT=60
DATA="/AdvWorks/Controls/billboard.ods"
CLASSID="clsid:6059B947-EC52-11CF-B509-00A024488F73">
  </OBJECT>
<%
  Else
    Set Ad = Server.CreateObject("MSWC.Adrotator")
    Response.Write(Ad.GetAdvertisement("/AdvWorks/adrot.txt"))
  End If
%>
```

Example 4. Example of the Browser Capabilities Component.

A Web user is not able to view the actual source code for Active Server Pages. All the browser sees is the HTML output of the ASP page.

The billboard rotator component simplifies the process of displaying different advertisements or announcements by managing air-time rotation for the different images. It allows a list of different advertisements to be assigned relative display-priority percentages. Every time the ASP Files are requested, the component can be used to display an ad based on the preset criteria.

HTTP request and response messages can have two parts: a header and a body. The following illustration shows these two parts.

{ewc MVIMG, MVIMAGE,!W06G010.bmp}

The header contains one or more header fields. Each field contains a single line of text that ends with a carriage return or a line feed (CR/LF) character pair.

The body of the message, if there is one, contains information sent by the user or the Web server.

For example, a GET request message does not contain a body, but the HTTP response message does. The body of the HTTP response message will contain a copy of the requested page.

The following table lists the types of header fields, the types of messages that can be contained in each header field, and a description of the message.

Header field	Message type	Description
Content-Type	Request and Response	The media type contained in the body.
Date	Request and Response	The date and time the message was generated.
Expires	Response	The date and time the content should be considered obsolete.
From	Request	The Internet e-mail address of the user running the browser.
If-Modified-Since	Request	Used with the GET request method to make it conditional. The page is returned only if it has been modified after the specified date.
Location	Response	The absolute URL of the page.
Refer	Request	The URL initiating the request.
User-Agent	Request	Information about the client software initiating the request.

For more information about header fields, go to the HTTP Specification Web site by clicking this icon. {ewc mvimg, mvimage,!intjump.bmp}

HTTP Request Messages

Each HTTP message contains an element that uniquely identifies it. For an HTTP request message, this identifier is the method line (also referred to as the method field).

A request method line has the following basic syntax:

```
HTTP-method   resource-identifier   HTTP/version
```

For example, the URL http://www.company.com/default.htm might generate this method line:

```
GET default.htm HTTP/1.0
```

The resource identifier is the requested file. The domain name is stripped from the resource identifier because when the HTTP request message is generated, a TCP/IP communication session will already have been established with the Web server.

The HTTP method communicates to the Web server the type of action the user is requesting. There are eight types of HTTP methods.

The following table lists four of the HTTP methods.

HTTP method	Description
GET	Retrieves the specified URL. This is the default method of a request.
POST	Sends data to a URL.
PUT	Creates or updates the file in a URL.
HEAD	Retrieves only the HTTP header information.

With the GET method, any information will be appended to the HTTP request for a page and sent in the message header. The size of the information sent with the GET method is limited to 1024 characters.

With the POST method, any data is sent in the body of the HTTP request message.

HTTP Response Messages

The header of a response message is composed of a status line and any additional response header fields.

If there is a body section, it follows after a blank line.

Click here to connect to the Microsoft Visual Basic Home Page on the Microsoft Web site:
{ewc mvimg, mvimage,!intjump.bmp}

This site provides information about Microsoft Visual Basic and links to numerous other pages for related information.

A Web application can have one Global.asa file. This file is stored in the virtual root of the application.

The Web server processes the Global.asa file either when it receives the first user request for an .asp file after start-up, or when a user without a session requests an .asp file.

You can include the following information in a Global.asa file:

® Application start events, end events, or both.

® Session start events, end events, or both.

® <OBJECT> tags that create components with application or session scope.

For information about server components, see Using ActiveX Server Components.

If you use the Web Project Wizard to create your Web application, Visual InterDev will create a Global.asa file with event procedure templates and comments to help you use this file.

Application Object Events

The **Application** object has the two events Application_OnStart and Application_OnEnd. You can add script to these events in the Global.asa file.

Any script you add to the Application_OnStart event will run when the application starts. Conversely, any script you add to the Application_OnEnd event will run when the application ends.

The following example code shows how to add the Application_OnStart event to the Global.asa file:

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Application_OnStart
    'Your Code Here
End Sub
</SCRIPT>
```

Session Object Events

Similar to the **Application** object, the **Session** object has the events Session_OnStart and Session_OnEnd. Any script you add to the Session_OnStart event will run when a user without an existing session requests an .asp file from your application. Any script you add to the Session_OnEnd event will run when a user session ends.

You can use the Session_OnStart event to direct users to a page that logs on to your site, regardless of which Active Server Page they request from your Web application.

To see a code sample that directs users to a login page, click this icon.
{ewc mvimg, mvimage,!code.bmp}

You can also use the Session_OnStart event to save session-specific information, such as information about a data connection.

To see a code sample that uses the Session_OnStart event to create **Session** object values for a data connection, click this icon.
{ewc mvimg, mvimage,!code.bmp}

To see a demonstration of how to use events in the Global.asa file, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Using the <OBJECT> Tag in the Global.asa File

You can use the <OBJECT> tag in the Global.asa file to create a component that runs on the Web server.

To use the <OBJECT> tag in a Global.asa file, you set the RUNAT attribute to Server, and the SCOPE attribute to Application or Session. To specify the component, you can use either its registered name, PROGID, or its registered number, CLASSID.

The following example code uses the PROGID to create a session-scope instance of the **Advertisement Rotator** component:

```
<OBJECT RUNAT=Server SCOPE=Session ID=MyAd PROGID="MSWC.Adrotator">
</OBJECT>
```

In the following example code, the registered number (CLASSID) method is used to create an application-scope instance of the **Advertisement Rotator** component:

```
<OBJECT RUNAT=Server SCOPE=Application ID=MyAd
CLASSID="Clsid:00000293-0000-0010-8000-00AA006D2EA4"></OBJECT>
```

When you use the <OBJECT> tag to declare a session-scope or application-scope instance of a component, the variable you assign to the component is stored in the session or application namespace. You do not need to use the **Session** or **Application** objects to access the instance of the component.

The following example code opens the instance of the **Advertisement Rotator** component that has been declared in the previous example code:

```
<%= MyAd.GetAdvertisement("addata.txt") %>
```

The following topics are covered in this white paper:

Article ID: Q157959
Creation Date: 21-OCT-1996
Revision Date: 10-DEC-1996
The information in this article applies to:

® Microsoft Visual C++, 32-bit Edition, versions 4.0, 4.1, 4.2

® Microsoft ActiveX SDK 1.0

Summary

ActiveX Controls are embedded in Web pages using the <OBJECT> tag. The CODEBASE parameter of the <OBJECT> tag specifies the location from which the control can be downloaded. CODEBASE can point at a number of different file types successfully:

® CODEBASE can point directly at an .ocx file:

```
CODEBASE="http://www.somesite.com/somecontrol..ocx#version=4,70,0,1086"
```

This solution relies on any supporting DLLs already being on the client machine. For Internet Explorer 3.0 and MFC ActiveX Controls built with Visual C++ 4.0 and 4.1, this is a good solution. Internet Explorer 3.0 ships with the supporting DLLs for Visual C++ 4.0 and 4.1 controls. If another Internet browser that is ActiveX-Control capable is used to view this control, this may not be the best solution.

® CODEBASE can point to an .inf file

```
CODEBASE="http://www.notasafesite.com/doyoutrustme.inf
```

An .inf file will control the installation of an .ocx file and its supporting files. This method is not recommended because it is not possible to sign an .inf file. Please see the REFERENCES section below for references on code signing.

® CODEBASE can point to a cabinet file:

```
CODEBASE="http://www.somesite.com/acontrol.cab#version=1,2,0,0"
```

Note that the #Version information applies to the control specified by the CLASSID parameter of the <OBJECT> tag.

This is the recommended way to package MFC ActiveX Controls and is the focus of this article. Packaging an MFC ActiveX Control in a cabinet file allows an .inf file to be included to control installation of the ActiveX Control, allows dependent DLLs to be named and a source provided, allows code signing, and automatically compresses the code for quicker download.

More Information

Cab Packaging Overview

The CABinet Development Kit contains the necessary tools to construct cabinet files. The kit is available by clicking the icon below:
{ewc mvimg, mvimage,!intjump.bmp}

The cabinet file pointed to by CODEBASE should contain the .ocx file and an .inf file that will control the installation of the ActiveX Control. Dependent DLLs that may already exist on the system should not be included in this cabinet file. The MFC DLLs are such a case: they should be packaged in a separate cabinet file and referred to by the controlling .inf file.

The following example illustrates how to package the MFC Spindial sample control:

The <OBJECT> tag to include the Spindial control in a Web page will look similar to the following:

```
<OBJECT ID="Spindial1" WIDTH=100 HEIGHT=51
  CLASSID="CLSID:06889605-B8D0-101A-91F1-00608CEAD5B3"
  CODEBASE="http://yoursite/spindial.cab#Version=1,0,0,001">
    <PARAM NAME="_Version" VALUE="65536">
```

```
    <PARAM NAME="_ExtentX" VALUE="2646">
    <PARAM NAME="_ExtentY" VALUE="1323">
    <PARAM NAME="_StockProps" VALUE="0">
    <PARAM NAME="NeedlePosition" VALUE="2">
</OBJECT>
```

In this case, Spindial.cab contains two files: Spindial.ocx and Spindial.inf. The command to build this cabinet file is similar to the following, depending on the path to your installation of the CABinet Development Kit:

```
C:\CabDevKit\cabarc.exe -s 6144 N spindial.cab spindial.ocx spindial.inf
```

The "-s 6144" parameter passed to Cabarc.exe reserves space in the cabinet for code signing.

The following is an example .inf file for the MFC Spindial control. This .inf file can be modified to download any MFC ActiveX Control by changing Spindial s information to the desired MFC ActiveX Control s information. See comments below.

```
; ========================= spindial.inf ========================

; This inf file controls the installation of the MFC Spindial
; control.

; This control has been compiled with Visual C++ version 4.2. The
; FileVersion tags in the dependent DLLs section on this file reflect
; this requirement.

; note - keep comments in .inf files on separate lines

[version]
; version signature (same for both NT and Windows 95) do not remove
signature="$CHICAGO$"
AdvancedINF=2.0

[Add.Code]
spindial.ocx=spindial.ocx
; These are the necessary supporting DLLs for MFC 4.2 ActiveX Controls
msvcrt.dll=msvcrt.dll
mfc42.dll=mfc42.dll
olepro32.dll=olepro32.dll

; thiscab is a keyword that, in this case, means that spindial.ocx
; can be found in the same .cab file as this .inf file
; file-win32-x86 is an x86 platform specific identifier
; See the ActiveX SDK-ActiveX Controls-Internet Component Download-
; Packaging component code for automatic download

[spindial.ocx]
file-win32-x86=thiscab
; *** add your controls CLSID here ***
clsid={06889605-B8D0-101A-91F1-00608CEAD5B3}
; add your ocx s file version here
FileVersion=1,0,0,001
RegisterServer=yes

; dependent DLLs

[msvcrt.dll]
; This is an example of conditional hook. The hook only gets processed
; if msvcrt.dll of the specified version is absent on client machine.
```

```
FileVersion=4,20,0,6164
hook=mfc42installer

[mfc42.dll]
FileVersion=4,2,0,6256
hook=mfc42installer

[olepro32.dll]
FileVersion=4,2,0,6068
hook=mfc42installer

[mfc42installer]
file-win32-x86=http://activex.microsoft.com/controls/vc/mfc42.cab

; If dependent DLLs are packaged directly into the above cabinet file
; along with an .inf file, specify that .inf file to run as follows:
;InfFile=mfc42.inf

; The mfc42.cab file actually contains a self extracting executable.
; In this case we specify a run= command.
run=%EXTRACT_DIR%\mfc42.exe

; ====================== end of spindial.inf ======================
```

Note the location for the MFC 4.2 DLLs:

```
[mfc42installer]
file-win32-x86=http://activex.microsoft.com/controls/vc/mfc42.cab
```

The Mfc42.cab file is provided and signed by Microsoft. The versions and files contained within follow what is listed in the .inf file above:

```
[msvcrt.dll] - FileVersion=4,20,0,6164
[mfc42.dll] - FileVersion=4,2,0,6256
[olepro32.dll] - FileVersion=4,2,0,6068
```

A signed cab file for MFC 4.0 and 4.1 ActiveX Control dependencies will be provided at a later time and will be referenced in this article. It is not necessary to provide these files for controls running under IE 3.0. However, if you are running a browser other than Internet Explorer 3.0 that supports ActiveX Controls, you may need to provide these DLLs.

References

® ActiveX SDK, Packaging component code for automatic download

® ActiveX SDK, Safety API Reference

® ActiveX SDK, Signing with Microsoft Authenticode(TM) Technology

® ActiveX SDK, Cabinet (.cab) File Technology: Data Compression and Disk Layout.

Article ID: Q149054
Creation Date: 26-MAR-1996
Revision Date: 21-JUN-1996

The information in this article applies to:

® Enterprise Edition of Microsoft Visual Basic for Windows, 32-bit only, version 4.0

Summary

This article describes the types of rdoResultset cursors in RDO and when to use them.

Note The following text comes directly from the Visual Basic online Help. Because there are no hot-links to the online Help, you must search for information explicitly. Go into Help, click on the Search button, click on the Find tab, type in Cursors, and then select Choosing a cursor type.

More Information

Choosing the right cursor for an application impacts performance and resource management. Your choice of cursor depends on how many rows you intend to access, how you need to navigate through the result set, how membership is determined, and how you intend to update the data.

In many cases, use of the forward-only type result set is the best choice as it exposes only one row of the result set at a time and is far easier for RDO to create. However, it is not a cursor and does not permit access to more than one row at a time.

1. Server-Side Cursor Support

   An important aspect of keyset or dynamic cursors is where the keyset is created. If the server supports server-side cursors, as with Microsoft SQL Server 6.0, you can specify that the cursor keyset is created and maintained on the server. With client-side cursors, cursor keysets are download to the workstation and stored in local memory. To enable server-side cursors, set the rdoDefaultCursorDriver or CursorDriver property.

2. Selecting a Cursor Type

   To select a specific type of rdoResultset cursor, set the RemoteData control's ResultsetType property or the type argument of the OpenResultset method to:

Resultset type	Constant
Forward-only	(Default) rdOpenForwardOnly
Static	rdOpenStatic
Keyset	rdOpenKeyset
Dynamic	rdOpenDynamic

3. Available Cursor Types

   The following table summarizes the four types of rdoResultset cursors:

Attribute	Forward-only	Static	Keyset	Dynamic
Updatable	Yes (SS) No (CL)	No (SS) Yes (CL)	Yes	Yes
Membership	Fixed	Fixed	Fixed	Dynamic
Visibility	One row	Cursor	Cursor	Cursor
Move current row	Forward	Anywhere	Anywhere	Anywhere
Result of a join	Yes	Yes	Yes	Yes

Note CL indicates that support for this cursor is provided by the ODBC cursor library. SS indicates support by Microsoft SQL Server.

Choose the type of rdoResultset object to create by using the type argument of the OpenResultset method or the ResultsetType property of the RemoteData control. If a type is not specifierd, the RemoteData control creates a keyset-type rdoResultset. When using RDO to create rdoResultset objects, the default type is forward-only.

Supported Cursor Types

Not all data sources support every type of cursor. The following table summarizes which type of cursor is supported on several typical data sources and on the RemoteData control:

Data source	Forward-only	Static	Keyset	Dynamic
SQL Server 4.2	Yes	Yes/CL	No	No
SQL Server 6.0	Yes	Yes	Yes	Yes
Oracle 7.1	Yes	Yes/CL	No	No
RemoteData control	No	Yes	Yes/DD	No

Note CL indicates that support for this cursor is provided by the ODBC cursor library. DD indicates support is provided subject to support by the ODBC driver.

---

1. Cursors and the RemoteData Control

   Creating an rdoResultset and setting the Resultset property with this new object, sets the ResultsetType property of the RemoteData control to the Type property of the new rdoResultset.

Note When using forward-only, read-only result sets, the rdoConnection is held open until the last row of data is accessed. While this can provide performance improvements over other cursors, it can also tie up connection resources.

References

Visual Basic Online Help

Building Client/Server Applications with Visual Basic

Hitchhiker's Guide to Visual Basic and SQL Server
Microsoft Press
ISBN: 1-55615-906-4

This section explains how to digitally sign files with the Authenticode technology included in the Microsoft ActiveX SDK. These digital signatures associate a software vendor's name and unique public key with a file; thus assuring accountability and integrity.

Currently, Authenticode allows software vendors to sign:

® .exe files.

® .cab files.

® .ocx files.

® .class files.

This section includes the following topics:

This section is a general introduction to code signing. Later, we will discuss Microsoft's technology, called Authenticode, which helps developers to easily sign their code.

One of the larger questions facing the software industry is this: How can users trust code that is published on the Internet? Currently, most Web pages contain only static information, but soon they will be filled with controls and applications that are downloaded and run locally, on the user's computer.

Packaged software uses branding and trusted retail channels to assure users of its integrity, but these are not available when code is transmitted on the Internet. Additionally, there is no guarantee that the code hasn't been altered while being downloaded. A more active approach must be taken to make the Internet a reliable medium for distributing software.

This section includes the following topics:

There are two issues that must be addressed:

® Ensuring integrity, which means verifying that the code hasn’t been tampered with since it was published.

® Ensuring accountability, which means informing users as to who published the code.

Microsoft's solution to these issues is Authenticode coupled with an infrastructure of trusted entities. We will discuss the infrastructure later in this document, when we explain certification authorities. Authenticode, which is based on industry standards, allows developers to ensure authenticity and integrity for their code through the use of *digital signatures*.

You use digital signatures when you have data you want to distribute, and you want to assure the recipients that it does indeed come from you and that it hasn't been tampered with. Signing data does not alter it. It simply generates a digital signature string you can bundle with the data.

Digital signatures are created using a public-key signature algorithm such as the RSA public-key cipher. A public-key algorithm actually uses two different keys: the *public key* and the *private key*. (These are called a key pair.) The private key is known only to its owner, while a public key can be available to anyone. Public-key algorithms are designed so that if one key is used for encryption, the other is necessary for decryption. Furthermore, the decryption key cannot reasonably be calculated from the encryption key. In digital signatures, the private key generates the signature, and the corresponding public key validates it.

In practice, public-key algorithms are often too inefficient for signing long documents. To save time, digital signature protocols use a cryptographic digest, which is a one-way hash of the document. The hashed document is signed instead of the document itself. Both the hashing and digital signature algorithms are agreed upon beforehand. Here is a summary of the process:

1. A one-way hash of the document is produced.
2. The hash is encrypted with the private key, thereby signing the document.
3. The document and the signed hash are transmitted.
4. The recipient produces a one-way hash of the document.
5. Using the digital signature algorithm, the recipient decrypts the signed hash with the sender's public key.

If the signed hash matches the recipient's hash, the signature is valid and the code is intact.

When code is associated with a publisher's unique signature, distributing software on the Internet is no longer an anonymous activity. Digital signatures assure accountability, just as a manufacturer's brand name does on packaged software. If an organization or individual wants to use the Internet to distribute software, they should be willing to take responsibility for that software. This approach is based on the premise that accountability is a deterrent to the distribution of harmful code.

A certificate is a set of data that completely identifies an entity, in this case a software publisher. The certification authority (CA) issues the certificate only after it has verified the entity’s identity. The data set includes the entity's public cryptographic key. When the sender of a message signs the message with its private key, the recipient of the message can use the sender's public key (retrieved from the certificate either sent with the message or possibly available elsewhere in the directory service) to verify that the sender is who it says it is.

One of the primary goals of a digital certificate is to confirm that the public key contained in a certificate is, in fact, the public key belonging to the person or entity to whom the certificate is issued. For example, a CA may digitally sign a special message (the certificate information) containing the name of some user, say "Alice," and her public key in such a way that anyone can verify that the certificate information message was signed by no one other than the CA and thereby convey trust in Alice's public key.

The typical implementation of digital certification involves a signature algorithm for signing the certificate. The process goes something like this:

1. Alice sends a certification request containing her name and her public key to a CA.
2. The CA creates a special message m from Alice's request, which constitutes most of the data in the certificate. The CA signs the message with its private key, obtaining a separate signature sig in the process. Then the CA returns the message m and the signature sig to Alice; the two parts together form a certificate.
3. Alice sends the certificate to Bob to convey trust in her public key.
4. Bob verifies the signature sig using the CA's public key. If the signature is verified, he accepts Alice's public key.

As with any digital signature, anyone can verify, at any time, that the certificate was signed by the CA, without access to any secret information. Bob only needs to get a copy of the CA's certificate in order to access the CA's public key.

A certificate is valid only for the period of time specified by the CA that issued the certificate. The certificate contains information about the beginning and expiration dates. The CA can also revoke any certificate it has issued and maintains a list of revoked certificates. This list is called a certificate revocation list (CRL), and is published by the CA so that anyone can determine the validity of any given certificate.

Certification authorities (CAs) are trustworthy persons or organizations that issue certificates to applicants, after verifying their identities. Certificates are verified through a hierarchy of these CAs. Each certificate is linked to the certificate of the CA that signed it. By following this hierarchy, or *verification path*, to a known, trusted CA, you can be assured that a certificate is valid. An example of this is illustrated in the following diagram.

{ewc mvimg, mvimage,!illust.bmp}
Sample Certification Hierarchy

In this example, Netwerks' certificate is certified by CA1 while Bob's is certified by CA3. CA1 has a certificate signed by CA2, so Netwerks can verify the CA1 certificate. CA2 also has a certificate signed by the root. CA3 (Bob's CA) has a certificate signed by the root. By moving up the verification chain to a common point (in this case, the root), Netwerks can verify Bob's certificate.

Certification authorities have two main duties:

® They publish the criteria for granting certificates.
® They grant certificates if an applicant meets the published criteria.

Other duties may include:

® Managing certificates (for example, enrolling, renewing, and revoking them).
® Storing root keys.
® Verifying evidence submitted by applicants.
® Providing tools for enrollment.
® Accepting the liability associated with these responsibilities.

To obtain a certificate from a CA, a software publisher must meet the criteria for either a commercial or an individual publishing certificate, and submit these credentials to either a CA. The criteria we will discuss are those proposed by Microsoft. Please note that standards bodies such as the World Wide Web Consortium are reviewing them and they are subject to change. We will then describe the overall process of obtaining a certificate for code signing.

Applicants for a commercial software publishing certificate must meet the following criteria:

- **Identification.** Applicants must submit their name, address, and other material that proves their identity as a corporate representative. Proof of identify requires either personal presence or registered credentials.
- **The pledge.** Applicants must pledge that they will not distribute software that they know, or should have known, contains viruses or would otherwise harm the user's computer or code.
- **Dun & Bradstreet rating.** Applicants must achieve a level of financial standing as indicated by a D-U-N-S number (which indicates a company's financial stability), and any additional information provided by this service. This rating identifies the applicant as a corporation that is still in business. (Other financial rating services are being investigated.) Corporations that do not have a D-U-N-S number at the time of application (usually because of recent incorporation) can apply for one and expect a response in less than two weeks.
- **Private key protection.** It is highly recommended that commercial applicants generate and store their private key using a dedicated hardware solution. This can be, for example, a magnetic stripe card, a plastic key with an embedded ROM chip (called a ROM key), or a smart card. For more information about storing keys, see Section 8.7 of Bruce Schneier's book, *Applied Cryptography*.

Two immediate questions are how do large software publishers determine who should apply for certificates and who should sign code? The answer depends on how the software publisher wants to control distribution of software on the Internet.

In a centralized approach, where the company wants total control of what code is published, there may be only one certificate, and strict guidelines for releasing code through one source. Other software publishers may allow each division, or even smaller groups or individuals within the company, to sign their own code using the corporate name. The point is that the software publisher must decide who can apply for a certificate and sign code and who takes responsibility for any code signed using certificates that bear the corporate name.

Using the Dun & Bradstreet rating as a criterion draws a line between "commercial" and "individual" developers. The intended distinction is between commercial persons or entities (that is, sole proprietors, partnerships, corporations, or other organizations that develops software as a business) and non-commercial persons or entities (that is, individuals or nonprofit corporations).

Applicants for an individual software publishing certificate must meet the following criteria:

- **Identification.** Applicants must submit their name, address, and other material that will be checked against an independent consumer database to validate their credentials.
- **The pledge.** Applicants must pledge that they cannot and will not distribute software that they know, or should have known, contains viruses or would otherwise maliciously harm the user's computer or code.

The value of an individual software publishing certificate is in the information it provides to users so they can decide whether or not to download the code. Knowing who authored the code, and that the bits have not been altered from the time the code was signed to the present, is reassuring information. Additionally, a browser could provide links to a publisher's Web pages so the user can obtain detailed information about the signed code, the author, and the certificate authority. After learning about this code and the author, the user may decide to run the code, or all future code, coming from this particular individual.

These are the steps to apply for and grant a certificate:

1. **Apply for a software publishing certificate**

   A software publisher's request for a certificate is sent to the CA. It is expected that CAs will have Web sites that step the applicant through the application process. Applicants will be able to look at the entire policy and practices statements of the CA. The utilities an applicant needs to generate signatures, such as Microsoft's Authenticode, should also be available.

   The applicant must generate a key pair using either hardware or software encryption technology. The public key is sent to the CA during the application process. For individuals, all of the necessary information can be transferred on-line. For commercial publishers, because of the identity requirements, proof of identification must be sent by mail or courier.

2. **Verify the applicant's credentials**

   The CA will examine the evidence to verify an applicant's credentials. To do this, they may employ external contractors such Dun & Bradstreet.

3. **Generate and issue the software publisher X.509 certificate**

   After the CA has decided that the applicant meets the policy criteria, it generates a Software Publisher Certificate (SPC) that conforms to the industry standard X.509 certificate format with Version 3 extensions. This certificate, which is distributed in the digital signature for the software, identifies the publisher, contains the publisher's public key, and is used to verify that the file has not been modified since it was signed. It is stored by the CA for reference and a copy is returned to the applicant via electronic mail.

   The publisher should review the contents of the certificate and verify that the public key works with the private key. After accepting the certificate, the publisher should include a copy in all published software signed with the private key.

   Commercial developers can expect a response to their application in less than two weeks. While there is no limit to the number of certificates commercial software publishers can obtain, it is up to the publisher to determine who gets a certificate, and how code is signed and distributed.

4. **Distribute signed software**

   The publisher can now begin signing and distributing software on the Internet. Publishers use utility programs to sign the software they intend to publish. The utility programs use the private key to generate a digital signature on a digest of the binary file and create a signature file containing a PKCS #7 signed-data object. (For more information about PKCS #7, see the RSA specification listed in the "Suggested Reading" section of this document.) The PKCS #7 signed-data object also contains a copy of the X.509 software publisher certificate. For portable executable (PE) image format files, the PKCS #7 signature file contents are stored in the binary file itself, in an additional section.

This section demonstrates how to sign code by creating digital signatures and associating them with files using Authenticode, which is provided with the ActiveX SDK. As we have seen, creating a fully verifiable certificate may assume the existence of a complex hierarchy of CAs. For testing purposes *only*, we provide a root certificate and a root private key. If you are an independent software vendor, you must obtain a certificate from GTE, VeriSign, Inc., or another CA before you begin signing code.

Authenticode

Authenticode consists of programs to digitally sign files and programs to check that the files were, indeed, successfully signed. Before you begin, first check that the underlying CryptoAPI is running. To do this, type:

```
c:>api *
```

This should generate SUCCESS messages until it is stopped.

The programs are:

® MakeCert, which creates a test X.509 certificate.

® Cert2SPC, which creates a test SPC.

® SignCode, which uses the SPC to sign a file.

® PeSigMgr, which checks to see that the file was signed.

® ChkTrust, which checks the validity of the file.

® DumpCert, which dumps the contents of a certificate.

We will now discuss these programs in more detail.

This section includes the following topics:

X.509 Certificate

A cryptographic certificate that contains a vendor's unique name and the vendor's public key.

PKCS #7 Signed Data

A Public Key Certificate Standard #7 (PKCS #7) signed-data object encapsulates the information used to sign a file. Typically, it includes the signer's certificate and the root certificate.

Certification Authority (CA)

A trusted entity that makes a statement (represented by an X.509 certificate) about the authenticity of another certificate.

Cryptographic Digest

A one-way hash function that takes a variable-length input string and converts it to a fixed-length output string (called a cryptographic digest.) This fixed-length output string is probabilistically unique for every different input string and thus can act as a "fingerprint" of a file. When a file with a cryptographic digest is downloaded, the receiver re-computes the digest. If the output string matches the digest contained in the file, then the receiver has proof that the received file was not tampered with and is identical to the file originally sent.

Local Registration Authority (LRA)

An intermediary between a publisher and a CA. The LRA can, for example, verify a publisher's credentials before sending them to the CA.

Portable Executable (PE) Image

The standard Win32 executable format.

Software Publishing Certificate (SPC)

A PKCS #7 signed-data object containing X.509 certificates, without the PKCS header.

Trust Provider

The portion of the operating system that decides whether or not a given file is trusted. This decision is based on the certificate associated with the file.

WIN_CERTIFICATE

A Win32 data structure that contains either a PKCS #7 signed-data object or an X.509 certificate.

To use Authenticode, the following files are required:

® Wintrust.dll (installed in the System\System32 directory)
® Digsig.dll (installed in the System\System32 directory)
® Signcode.dll (installed in the System\System32 directory)
® Makecert.exe (creates an X.509 certificate for testing purposes only)
® Cert2SPC.exe (creates an SPC for testing purposes only)
® SignCode.exe (digitally signs code)
® PeSigMgr.exe (checks that SignCode was successful)
® ChkTrust.exe (checks the validity of the file)
® DumpCert.exe (dumps the contents of a certificate)
® Root.cer (a root certificate for testing purposes only)

The X.509 protocols include a structure for public-key certificates. A CA assigns a unique name to each user and issues a signed certificate containing this name and the user's public key. The following diagram shows an X.509 certificate.

{ewc mvimg, mvimage,!illust.bmp}
X.509 Certificate

These are the meanings for each field.

Field	Meaning
Version	Identifies the certificate format.
Serial Number	Is unique to the CA.
Algorithm Identifier	Identifies the algorithm used to sign the certificate, together with any necessary parameters.
Issuer	The name of the CA.
Period of Validity	A pair of dates. The certificate is valid during the time period between the two.
Subject	The name of the user.
Subject's Public Key	Contains the public key algorithm name, any necessary parameters, and the public key.
Signature	The CA's signature.

The topic of digital signing is discussed more fully in the following documents.

CCITT, Recommendation X.509, *The Directory-Authentication Framework*, Consultation Committee, International Telephone and Telegraph, International Telecommunications Union, Geneva, 1989.

Microsoft Cryptographic Service Provider Programmer's Guide, Microsoft, 1995.

Microsoft Application Programmer's Guide, Microsoft, 1995

RSA Laboratories, *PKCS #7: Cryptographic Message Syntax Standard*. Version 1.5, November, 1993.

Schneier, Bruce, *Applied Cryptography*, 2d ed. New York: John Wiley & Sons, 1996.

http://www.microsoft.com/intdev/security

http://www.rsa.com

Although support for cookies has been adopted by the major browser manufacturers, two problems remain:

- Cookies are not a standard feature of HTTP or other specifications. Therefore, they may not be available for all browsers.
- Because cookies enable a Web server to create and edit a file on a user's computer (which is insecure), both Netscape and Microsoft enable users to turn off this feature.

To prevent the Web server from sending the HTTP response to the user until all server-side script on the current Active Server Page has been processed, you can buffer the content of the response message.

Note Waiting for the server to finish processing all server-side script may cause a lengthy delay. To send pieces of the response to the user, you can use the **Flush** method of the **Response** object throughout your code.

Setting the Buffer Property

To enable buffering, you set the **Buffer** property to **True**, as shown in the following example code:

```
Response.Buffer = True
```

Note You cannot set the **Buffer** property after the server has sent output to the user. For this reason, you should set the **Buffer** property in the first line of the .asp file.

Handling Errors with Buffering

If an error occurs during processing, you can use the **Redirect** method of the **Response** object, with buffering enabled. First, you clear the buffer with the **Clear** method, and then you use the **Redirect** method.

When an error occurs, the following example code will clear the buffer and redirect the user to an error page:

```
Response.Buffer = True
On Error Resume Next
'code that may cause an unrecoverable error,
'such as failing to open a data connection
If Err.number <> 0 Then
    Response.Clear
    Response.Redirect "error.htm"
End If
```

Article ID: Q162840
Creation Date: 29-JAN-1997
Revision Date: 03-FEB-1997

The information in this article applies to:

® Microsoft Internet Information Server, version 3.0

® Microsoft Active Server Pages, version 1.0

Summary

ADO.Connection was changed to ADODB.Connection in the final released version of Microsoft Active Server Pages, version 1.0.

More Information

Scripts written using pre-released versions of the Microsoft Active Server Pages require an update to all scripts that use ADO.Connection. For example, the following:

```
<% Set obectvar = Server.CreateObject("ADO.Connection") .........%>
```

must now be changed to the following:

```
<% Set obectvar = Server.CreateObject("ADODB.Connection") .........%>
```

References

For additional information on the proper use of ADODB.Connection, please see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q158737**
TITLE: How To Create a Simple Query in an ActiveX Layout

Active Server Pages online documentation,
http://localserver/IASDocs/ASPDocs/roadmap.asp.

KBCategory: kbprg kbnetwork
KBSubcategory: AXSFHTML AXSFCompADO AXSFDataBase
Additional reference words: 1.00

Article ID: Q163009
Creation Date: 31-JAN-1997
Revision Date: 17-FEB-1997

The information in this article applies to:

® Microsoft Active Server Pages, version 1.0

Summary

The version of VBScript (VBS) provided with Active Server Pages currently does not have access to the values of constants defined by the Scripting object. As a result, you must provide a definition of the constants you wish to use in your scripts. Because the documentation does not provide thevalues for these constants, it can be difficult to determine what the numerical value is for a given constant.

Consider the following code example:

```
Set fs = Server.CreateObject("Scripting.FileSystemObject")
Set a = fs.OpenTextFile("c:\testfile.txt", ForAppending, FALSE)
```

If you cut and paste this code into an .asp file, you get the following error because VBS doesn't recognize 'ForAppending':

```
Microsoft VBScript runtime error '800a0005'
Invalid procedure call or argument: 'fs.OpenTextFile'
```

The best approach is to use a server-side include file to provide a definition of the constants for your script. This is similar to using the Adovbs.inc or Adojavas.inc files provided with ADO.

More Information

Here is a list of the values of the most commonly used constants for theScripting object:

```
iomode
======
ForAppending = 8
ForReading = 1
ForWriting = 2

format
=====
TristateFalse = 0
TristateMixed = -2
TristateTrue = -1
TristateUseDefault = -2
```

These are determined by viewing the Microsoft Scripting Runtime type library, contained in Scrrun.dll, in the Visual Basic Object Browser.

To provide an include file that defines these values, paste the text below into a file called Filevbs.inc, and save it to a virtual root. Then add an include directive, similar to the one below, to the ASP file that needs to access the definitions of these constants.

```
<!--#include virtual="/ASPSAMP/SAMPLES/FILEVBS.INC"-->
```

(Save the text below to file: Filevbs.inc)

```
<%
   '--------------------------------------------------

   ' FileSystemObject constants include file for VBScript
   '
```

```
'-------------------------------------------------

'---- iomode Values ----
Const ForAppending = 8
Const ForReading = 1
Const ForWriting = 2

'---- format Values ----
Const TristateFalse = 0
Const TristateMixed = -2
Const TristateTrue = -1
Const TristateUseDefault = -2

%>
```

KBCategory: kbtool kbprg kbdocerr
KBSubcategory: AXSFVBS
Additional reference words: 1.00

Article ID: Q163501
Creation Date: 11-FEB-199
Revision Date: 21-FEB-1997
The information in this article applies to:

® Microsoft Active Server Pages, version 1.0

Symptoms

If you attempt to access Active Server Pages(ASP) files located in subdirectories under a virtual root, which have restricted access permissions, the server may return the following error:

HTTP/1.0 Invalid Application Name

Cause

This message is caused by a problem in Active Server Page's handling of Access Control Lists (ACLs).

Resolution

To resolve this problem, either allow FULL CONTROL to the directory encountering the problem or apply the patch available for Active Server Pages.

Status

Microsoft has confirmed this to be a problem in Active Server Pages version1.0.

A supported fix is now available, but is not fully regression-tested and should be applied only to systems experiencing this specific problem. Unless you are severely impacted by this specific problem, Microsoft recommends that you wait for the next Service Pack that contains this fix. Contact Microsoft Product Support Services for more information.

KBCategory: kbusage kbfixlist kbbuglist
KBSubcategory:
Additional reference words: 1.00

Article ID: Q163499
Creation Date: 11-FEB-1997
Revision Date: 11-FEB-1997

The information in this article applies to:

® Microsoft Active Server Pages, version 1.0

Summary

This article provides an example of how to simulate a dynamically growing form using Active Server Pages. This example allows the user to enter items one at a time, and add them to a list. Once the list has been created, it can be submitted as a whole.

More Information

In the following example, a basic form is drawn with one text box. The user can enter information in that text box, and press the 'Add Item to List' button. This button submits the form to the same file. As each item is entered, it is added to the form as a new text box with the name ITEMS. Ultimately when the whole list is submitted, there are several items in the list that can be looped through as a collection of ITEMS.

To test this technique, save the text below to a file named Dynaform.asp. Be sure that this file is located in an IIS Virtual Root.

File: DYNAFORM.ASP

```
<%@ language = vbscript%>
   <% Response.Expires = 0 %>
   <HTML>
   <HEAD>
   <TITLE>Dynamically Growing Form</TITLE>
   </HEAD>
   <BODY>
   <%
   If Request("Action") = "Submit the List" Then
     ' Show what was entered.
     Response.Write "<B>Here are the Items submitted:</B><BR>"
       nItems = Request.Form("Items").Count
       For i = 1 To nItems
         ' Show submitted Items
         Response.Write Request.Form("Items")(i) & "<BR>"
         Next
         Response.Write Request("Item") & "<BR>"
   Else
     ' Create the form from all items. %>
     <FORM Action=dynaform.asp Method=Post>
     <B>Items:</B><BR>
     <%
     nItems = Request.Form("Items").Count
     For i = 1 To nItems
       ' Show previously submitted Items
       Response.Write "<INPUT Type=Text Name=Items Value=""" & _
         Trim(Request.Form("Items")(i)) & """><BR>"
     Next

     If Request.Form("Item") <> "" Then
       ' paint a new input box, and store the old Item in Items collection
       Response.Write "<INPUT Type=Text Name=Items Value=""" & _
         Trim(Request.Form("Item")) & """><BR>"
```

```
      Response.Write "<P>Please enter an Item,<BR>"
      Response.Write "and submit them one at a time<BR>"
      Response.Write "by pressing the Add Item button.<BR>"
      Response.Write "<INPUT Type=Text Size=50 Name=Item Value="""""">"
    Else
      ' No Item was submitted, don't show an error
      Response.Write "<P>Please enter an Item,<BR>"
      Response.Write "and submit them one at a time<BR>"
      Response.Write "by pressing the Add Item button.<BR>"
      Response.Write "<INPUT Type=Text Size=40 Name=Item Value="""""">
      <BR>"
    End If
  %>
    <P>
    <INPUT Type="Submit" Name="Action" Value="Add Item to List">
    <INPUT Type="Submit" Name="Action" Value="Submit the List">
    <BR>
  <% End If %>

  </FORM>
  </BODY>
  </HTML>
```

KBCategory: kbprg kbcode kbhowto
KBSubcategory: AXSFHTML
Additional reference words: 1.00

Article ID: Q163133
Creation Date: 03-FEB-1997
Revision Date: 04-FEB-1997

The information in this article applies to:

® Microsoft Active Server Pages, version 1.0

Summary

The Microsoft Knowledge Base (KB) is categorized by using keywords. This article lists the category and subcategory keywords specific to articles in Microsoft Active Server Pages (ASP).

More Information

Microsoft ActiveX SDK KBSubcategory Keywords

Each KB article in the Active Server Pages collection may contain one or more product-specific subcategory keywords (called KBSubcategory keywords) that place the article in an appropriate Active Server Pages category. For example, you can find all the Browser Component articles by using AXSFCompBrowser as the keyword when you query the Microsoft Knowledge Base.

An article usually has only one subcategory keyword, but it may have more.

Following are the topics and the corresponding KBSubcategory keywords.

Topic	KBSubcategory Keyword
AXSFCompAdRot	Ad Rotator Component
AXSFCompAll	All Components
AXSFCompBrowser	Browser Component
AXSFCompContLink	Content Linking Component
AXSFCustControl	Custom Control
AXSFCompADO	Database Connection Component
AXSFDataBase	Database Issues
AXSFHTML	General HTML Issues
AXSFJScript	JScript-Related Issues
AXSFSQL	SQL Issues
AXSFCompTextStream	TextStream Component
AXSFVBS	VBS-Related Issues

Product-Specific Keywords

You can use the KBSubcategory keywords to organize Active Server Pages articles or to search for specific groups of Active Server Pages articles.

For information about KBSubcategory keywords for other Microsoft developer products, please query the Microsoft Knowledge Base using these keywords:

```
dskbguide and kbkeyword
```

KB-Wide Keywords

Each article in the Active Server Pages collection also contains at least one generic, KB-wide category keyword (called a KBCategory keyword). The KBCategory keywords are standard throughout the Microsoft Knowledge Base, appearing in all KB articles, regardless of product. You can use the KBCategory keywords to organize all KB articles or to search for articles across several Microsoft products.

References

For more information about KBCategory keywords, please see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q94671**
TITLE: Categories and Keywords for All Knowledge Base Articles

KBCategory: kbref kbkeyword kbhowto
KBSubcategory:
Additional reference words: 1.00

```
'when application starts, read hit counter information
'from a text file
Sub Application_OnStart
    Dim fsVisitors   'FileSystemObject object
    Dim fileVisitors 'TextStream object

    Set fsVisitors = Server.CreateObject("Scripting.FileSystemObject")
    Set fileVisitors = fsVisitors.OpenTextFile("c:\visitors.txt")
    'Read counter value from text file
    Application("visitors") = fileVisitors.ReadLine
    fileVisitors.Close
End Sub

'when application ends, save hit counter in a text file
Sub Application_OnEnd
    Dim fsVisitors   'FileSystemObject object
    Dim fileVisitors 'TextStream object

    Set fsVisitors = Server.CreateObject("Scripting.FileSystemObject")
    Set fileVisitors = fsVisitors.CreateTextFile("c:\visitors.txt", True)
    'Write counter value to text file
    fileVisitors.Writeline(Application("visitors"))
    fileVisitors.Close
End Sub

'when session starts, increment the hit counter
Sub Session_OnStart
    Application.Lock
    'Increment counter
    Application("visitors") = Application("visitors") + 1
    Application.Unlock
End Sub
```

For an introduction to this chapter, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

In Chapter 1: Planning a Web Site, you learned the importance of designing a Web site for three different logical entities: user services, business services, and data services. In this chapter, you will learn how to create business services that will run on a Web server.

You will learn how to use Visual Basic version 5.0 to build ActiveX server components that contain business rules. You will also learn how to call these ActiveX server components from a page on your Web site.

The following illustration highlights the technologies you will learn about in this chapter.

{ewc MVIMG, MVIMAGE,!w08g070.bmp}

Objectives

By the end of this chapter, you will be able to:

® List two benefits of creating business objects.

® List the advantages of creating business objects as ActiveX server components.

® Build an ActiveX server component with Visual Basic 5.0.

® Call an ActiveX server component from an Active Server Page.

The most common way to use business processes is to model them after objects This approach enables you to represent and implement business processes in an object-oriented environment as you plan a Web site.

There are several advantages to using an object-oriented model to create business processes.

® Simplifying Complex Processes

When you first look at creating a business process, the pieces of the process may seem very complex. To simplify a business process, you can break each part of the process into smaller pieces, or objects. You can then easily conceptualize the process by thinking of each part as a specific object, where objects perform specific functions of the process.

® Interoperability

With objects, each process is more accessible to the server. For example, you can use an object from a Visual Basic application and from an Active Server Page. The various parts of a business process can access other business processes in a consistent manner.

® Encapsulation

Each business object encapsulates the data and functionality needed to accomplish its task. A business object exposes only those methods needed by other business objects. By encapsulating only specific data and functionality in each object, business objects are self-contained and isolated from changes made to other objects.

In this section, you will learn how to test a Visual Basic project to make it available to other applications. You will also learn to how to call ActiveX Server components from an .asp file and from the Advanced Data Connector.

This section includes the following topics:

A class module is a type of code module that Visual Basic provides. Each class module defines one type of object.

Class modules represent a logical object in your component. In an application, you can have several class modules. At run time, you create an object by creating an instance of a class. A class is a template for an object. An object is an instance of a class.

For example, you can create an **Employee** class that has properties such as **LastName** and **FirstName**, and a method such as **Hire**.

Adding a Class Module to a Project

When you create a new ActiveX DLL project in Visual Basic, it creates project with one class module.

To see an animation of how to create an instance of a class, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

You can add a new class module by clicking **Add Class Module** on the Visual Basic **Project** menu. You can then add methods, properties, and events to the class.

A class module is a template for an object. An object is an instance of a class. To create an instance of a class in the project that defines the class, you use the **Dim** and **Set** statements. This code sample creates an instance of the Math class.

```
Dim Demo1 As Math
Set Demo1 = New Math
```

You can also use the more compact syntax such as **Dim MyObject1 As New Math** instead of using the **Set** statement.

Once you have an object created, you can use methods and properties of the object. This example code invokes the **SquareIt** method of the Demo1 object.

```
i = Demo1.SquareIt (Num:=5)
```

Class modules have two built-in events: Initialize and Terminate.

To add code to a class module event, you open a Visual Basic code window for the class, and then click **Class** in the **Object** drop-down list box.

Initialize Event

The Initialize event occurs when an instance of a class is created, but before any properties have been set. You use the Initialize event to initialize any data used by the class, as shown in the following example code.

```
Private Sub Class_Initialize()
    'Initialize data.
    iDept = 5
End Sub
```

Terminate Event

The Terminate event occurs when an object variable goes out of scope or is set to **Nothing**. You use the Terminate event to save information, unload forms, or perform tasks that you want to occur only when the class terminates, as shown in the following example code.

```
Private Sub Class_Terminate()
    'Any termination code.
End Sub
```

For information about adding code to the Initialize and Terminate events, search for "Coding Robust Initialize and Terminate Events" in Visual Basic Books Online, and then click "Adding Classes to Components."

To add methods to an object, you create public Sub or Function procedures within the class module for that object. The public Sub and Function procedures will be exposed as methods for the object.

Note You must add the **Public** keyword before Sub or Function to make the procedure available to other objects and applications. Otherwise, the procedure is private and can be accessed only from the object.

To create a method for an object, you can either type the procedure heading directly in the code window or click **Add Procedure** on the **Tools** menu and complete the dialog box.

To see a demonstration of how to add a method to an object, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

This example code creates a method that accepts a number and then returns the number squared.

```
Public Function SquareIt (Num As Integer) _
    As Integer
    SquareIt = Num * Num
End Function
```

To view the properties and methods you have defined for an object, you can use the Object Browser. For information about using the Object Browser, see "Object Browser" in Visual Basic Help.

In this section, you will learn how to create a new ActiveX DLL project in Visual Basic 5.0.

You will learn how to add class modules to the project to create objects that encapsulate the internal functionality of an ActiveX Server component. You will also learn how to expose methods from objects to make the ActiveX Server component interoperable with other components and applications.

This section includes the following topics:

When you create a new ActiveX DLL or ActiveX EXE project with Visual Basic, you also set a number of other properties that affect how your code component will run.

Project Properties

In Visual Basic, you set properties for a project by clicking **Project *ProjectName* Properties** on the **Project** menu. Then, click the **General** tab of the **Project Properties** dialog box to select the options you want.

Project Type

The **Project Type** field provides the four template options: Standard EXE, ActiveX EXE, ActiveX DLL, and ActiveX Control. When you create a new ActiveX DLL or ActiveX EXE project, Visual Basic automatically sets the **Project Type** property.

The project type determines how some of the other project options can be set. For example, options on the **Component** tab are not available when the project type is set to Standard EXE.

Startup Object

For most ActiveX EXEs and ActiveX DLLs, you set the **Startup Object** field to **(None)**. If you want initialization code to run when the component is loaded, set the **Startup Object** property to **Sub Main**. If you want initialization code to run when an instance of a class is created, use the **Class_Initialize** event as explained in Adding Events to Class Modules in this chapter.

Project Name

The **Project Name** field specifies the component name to which the client application refers.

Project Description

The **Project Description** field enables you to enter a brief description of the ActiveX code component, along with the objects it provides.

The contents of this field will appear in the **References** dialog box when you select references for other Visual Basic projects. The text also appears in the **Description** pane at the bottom of the Object Browser.

Unattended Execution

The **Unattended Execution** check box specifies whether or not the component will be run without user interaction. Unattended components do not have a user interface. Any run-time functions, such as messages that normally result in user interaction, are written to an event log.

If this box is checked, the project will be built to use the threading apartment model. Using this model will make your objects more efficient when they run.

To determine the behavior of a component, you set properties for each class module in the component.

Name Property

To create a name for the class, set the **Name** property in the **Properties** dialog box. This name will be used by the client application to create an instance of an object.

For example, the following code, creates an instance of a class named **MyClass** which is defined in the project named **Project1**.

```
Dim x
Set x = CreateObject ("Project1.My Class")
```

Instancing Property

The **Instancing** property determines whether or not applications outside of the Visual Basic project that defines the class can create new instances of the class, and if so, how those instances are created.

The available **Instancing** property settings are different in ActiveX EXE and ActiveX DLL components, as shown in the following illustration.

{ewc MVIMG, MVIMAGE,!W08g050.bmp}

When you create a business object, set the **Instancing** property to MultiUse.

The following table defines each of the **Instancing** property settings.

Setting	Description
Private	Other applications are not allowed access to type library information about the class and cannot create instances of it. **Private** objects are used only within the project that defines the class.
PublicNotCreatable	Other applications can use objects of this class only if the component creates the objects first. Other applications cannot use the **CreateObject** method or the **New** operator to create objects of this class. Set the **Instancing** property to this value when you want to create dependent objects.
SingleUse	A instance of the component can only be used by one client. If multiple clients create objects from the component, a new instance of the component is created for each client.
GlobalSingleUse	Similar to SingleUse, except that properties and methods of a class can be invoked as though they were global functions.
MultiUse	Multiple clients can use the same instance of the component. If multiple clients create objects from the component, the same instance of the component is used for all clients.
GlobalMultiUse	Similar to MultiUse, except properties and methods of the class can be invoked as though they were global functions. It is not necessary to create an explicit instance of a class, because one will automatically be created.

You compile an ActiveX server component as either an EXE or a DLL, depending on the properties you have set in the **Project Properties** dialog box.

Type Library ID

When you compile an ActiveX EXE or ActiveX DLL component, Visual Basic creates a unique Type Library ID. This ID is entered into the system registry when you register the code component.

In the **Project Properties** dialog box, the **Project Compatibility** field is selected by default. This setting specifies that the same Type Library ID should be reused each time you compile a code component.

If you make any change to the component (for example, if you delete a property) that is incompatible with the version on your Web server specified in the **Project Compatibility** field, Visual Basic displays a warning message and generates a new Type Library ID.

To change the **Project Compatibility** field, click **Project Properties** on the **Project** menu. Click the **Component** tab, and then click **Project Compatibility**.

Before you can use an ActiveX Server component, it must be registered.

Registering a Component

The process to register a component differs for out-of-process and in-process components.

There are several ways to register an out-of-process (EXE) component:

® Run the component.

It automatically registers itself each time it runs.

® Run the component with the /Regserver command line argument.

The component ends immediately after registering itself.

® Create a Setup program.

When you run the Setup program, the component is registered.
{ewc mvimg, mvimage,!tip.bmp}

There are several ways to register an in-process (DLL) component:

® Run Regsvr32.exe.

Regsvr32 is a utility that will register a DLL. Pass the DLL file name as an argument to the Regsvr32 utility as shown in this example:

```
Regsvr32.exe mydll.dll
```

® Create a Setup program.

When you run the Setup program, the component is registered.

Unregistering the Component

When a component is no longer needed, it can be unregistered.

To remove an EXE entry from the registry, run the component with the /UnRegserver command line argument, as shown in the following code:

```
myServer.Exe /UnRegserver
```

To remove a DLL entry from the registry, run Regsvr32.exe, including the /u option and the name of the DLL file, as shown in the following code:

```
Regsvr32.exe /u mydll.dll
```

This topic discusses how to raise and trap run-time errors by using the **Raise** method of the **Err** object. It also details how options that you set for Break mode can affect how errors are handled when raising errors to the client.

Raising an Error in the Client

When you create an ActiveX component, you can provide error messages to the client application. To pass an error back to a client application, you use the **Raise** method in a component. For the error to be raised in the client, you must call **Raise** from your method's error-handling routine or with error handling disabled.

The **Raise** method has the following syntax.

ERR.Raise (*Number*, *Source*, *Description*, *HelpFile*, *HelpContext*)

The error number is generated by adding the intrinsic constant **vbObjectError** to the error number. The resulting number is returned to the client application. This ensures that your error numbers do not conflict with the built-in Visual Basic error numbers.

Breaking on Errors

You can change the way Visual Basic enters Break mode when an error occurs in your code component. To do this, you set **Error Trapping** options on the **General** tab of the **Options** dialog box. It is important to be aware of these options, because they can cause your application to break on an error, even if you have created an error handler.

Visual Basic provides three options for setting breakpoints, as shown in the following table.

Option	Description
Break on All Errors	Any error causes the project to enter break mode, whether or not an error handler is active, and whether or not the code is in a class module.
Break in Class Module	Any unhandled error that has been produced in a class module will cause the project to enter Break mode at the line of code in the class module that produced the error. When you debug an ActiveX server project by running an ActiveX client test program from another project, set this option in the ActiveX server project to break on errors in its class modules, instead of returning the error to the test program.
Break on Unhandled Errors	If an error handler is active, the error is trapped without entering Break mode. If there is not any active error handler, the error causes the project to enter Break mode. An unhandled error in a class module will cause the project to enter Break mode on the line of code that invoked the offending procedure of the class.

1. Which Visual Basic project template would you use to build an in-process ActiveX server component?

{ewc mvimg, mvimage,!answer.bmp}
A. ActiveX Control

{ewc mvimg, mvimage,!answer.bmp}
B. ActiveX DLL

{ewc mvimg, mvimage,!answer.bmp}
C. ActiveX EXE

{ewc mvimg, mvimage,!answer.bmp}
D. Standard EXE

2. Assume that a business process is used to take customer orders. Before placing the order, the Inventory table must be checked to make sure the item is in stock. If it is, a new record must be added to the Shipping table to make sure the item gets shipped. Which of the following strategies should you use to implement this business process?

{ewc mvimg, mvimage,!ans
A. Create a new ActiveX server component that implements the business process with an object named **Customer.Order**, and call it from an Active Server Page.

wer.bmp}

{ewc mvimg.mvimage.!answer.bmp}

B. Create a new Active Server Page that uses ActiveX Data Objects (ADO) to implement the business process.

{ewc mvimg.mvimage.!answer.bmp}

C. Create a stored procedure named customerorder that implements the business process, and call it from an Active Server Page.

{ewc mvimg.mvimage.!answer.bmp}

D. Create an .exe file as an ActiveX server component on the server to hold this business process as well as other processes.

3. How do you export a method from a class module in Visual Basic 5.0?

{ewc mvimg.mvimage.!answer.bmp}

A. Mark the Visual Basic project for unattended execution.

{ewc mvimg.mvimage.!answer.bm

B. Set the **Instancing** property to MultiUse.

p}

{ewc mvimg, mvimage,!answer.bmp}

C. Add the **Public** keyword before the method.

{ewc mvimg, mvimage,!answer.bmp}

D. Set the Visual Basic project type to ActiveX DLL.

4. What are all of the ways in which an in-process component can be registered?

{ewc mvimg, mvimage,!answer.bmp}

A. During Setup, by using the file RegSvr32.exe, and when you compile the component with Visual Basic 5.0.

{ewc mvimg, mvimage,!answer.bmp}

B. During Setup, and when you compile the component with Visual Basic 5.0.

{ewc mvimg, mvimage,!answer.bmp}

C. By using the file RegSvr32.exe, and when you compile the component with Visual Basic 5.0.

{ewc mvimg.mvimage.!answer.bmp}

D. During Setup, and by using the file RegSvr32.exe.

5. What is defined as a piece of heuristic logic that provides guidelines for how a piece of information should be processed?

{ewc mvimg.mvimage.!answer.bmp}

A. A business rule

{ewc mvimg.mvimage.!answer.bmp}

B. A business process

{ewc mvimg.mvimage.!answer.bmp}

C. An ActiveX server component

{ewc mvimg.mvimage.!answer.bmp}

D. A business object

In this section, you will learn how business rules in server components can increase the efficiency of your Web site. You will also learn about the attributes used to create business objects.

This section includes the following topics:

In this topic, you will learn about two basic concepts of business services: business rules and business processes. You can process a number of business rules together as part of a business service.

To see an illustration of how different business rules can be used together, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Business Rules

A business rule is a piece of heuristic logic that provides guidelines for how a piece of information should be processed. For example, if a credit card number is entered into a form, one business rule may check the credit card number for the limit available in that account. If the available credit limit exceeds the transaction, credit will be granted and the transaction will proceed.

Business Processes

A business process is a sequence of related tasks that produce a specific response to a user's request. For example, when a user submits an order form to purchase a product from an online catalog, a transaction is executed. This is a business process.

Other examples of business processes include activities such as opening a new bank account, retrieving customer information, or retrieving benefit options for a specific employee. For each of these examples, a business process acts on a business rule.

To see an illustration of how the business processes for the State University Web Site can be separated, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

With Visual Basic, you can build and run in-process or out-of-process code components. In-process components are ActiveX DLLs. Out-of-process components are ActiveX EXEs.

The following table shows the advantages and disadvantages of using in-process and out-of-process components.

Type of Component	Advantages	Disadvantages
In-process DLL	Faster access to objects	Less fault-tolerant. If the DLL fails, the entire executable fails.
Out-of-process EXE	Faults are limited to just the out-of-process EXE. If the EXE fails, other processes in the system will not fail	Are slower because of marshalling.

Business objects that are implemented as ActiveX server components are generally in-process components because they run faster.

For more information about the differences between in-process and out-of-process components, see "Building Code Components" in Visual Basic Books Online.

Project Templates

When you create a new project in Visual Basic, you choose a template on which to base the new project. To create a ActiveX server component, you choose either the ActiveX EXE or the ActiveX DLL template in the **New Project** dialog box. Selecting either type of template sets a number of default values that are important for creating code components.
{ewc mvimg, mvimage,!tip.bmp}

The following illustration shows the standard Visual Basic templates.

{ewc MVIMG, MVIMAGE,!W08g040.bmp}

To see a demonstration of creating a new project, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Once you have set up a test project and established a reference to the component, you can debug the component just as you would any Visual Basic application.

You can step between the client application and component, and use the standard debugging tools to set break points, step through code, use watch expressions, and so on.

Handling Missing References

In the course of debugging components, you may see the following error message: **Connection to type library or object library for remote process has been lost. Press OK for dialog to remove reference.**

This message indicates that the globally unique identifier (GUID) of the component's type library has changed, and the test project cannot locate the type library.

To clear the missing reference, click OK. In the **References** dialog box, the word **MISSING** will appear next to the name of the component. Clear the check mark next to the name of the missing component, and then click OK.

The first step in testing an ActiveX Server component is to create a test project that will use the component.

This topic explains how to test ActiveX DLL and ActiveX EXE Server components.

Testing an ActiveX DLL

To test an ActiveX DLL, you can add a Standard EXE project to the ActiveX DLL's project group, and test the in-process component within a single instance of Visual Basic. You can then step directly from the test component code to the in-process component code.

Testing an ActiveX EXE

To test an ActiveX EXE, you must run the component within one instance of Visual Basic, and run the test application in a second instance of Visual Basic. In this case, you cannot step directly from one component to the other, but you can use all of the other Visual Basic debugging tools.

Handling errors requires close communication between clients and components. If possible, a component and its clients should handle errors, rather than displaying a message to the user.

When a client application calls a method of an object provided by a component, the method can use the following two ways of providing error information.

® Raise an error that the client application must handle. In this case, the client uses conventional error-handling statements to handle errors that are raised.

–or–

® Return an error code that the client application must test for and handle.

Using a Raised Error

In a code component, the method that was called by the client application uses the **Raise** method of the **Err** object to raise an error to the client application. The client implements an error handler to trap the error.

Developers of client applications can either implement in-line error handling by using the **One Error Resume Next** statement, or by writing error-handling routines by using the **On Error Goto** statement.

Using a Return Value

In a code component, the method called by the client application returns a value to the client. The client examines the return value to determine if an error has occurred, and if so, which one.

With this type of error handling, developers of client applications must use in-line error handling. Code must always test the return value after calling a method.

For information about error handling in components and clients, search on **Debugging, Testing, and Deploying Components** in Visual Basic Books Online, and then click **Topics on Debugging**.

When you build a component and provide classes for use by clients, the compiler in Visual Basic automatically generates a type library. Client developers can use the Object Browser to see what functionality the component offers, and can connect to your component by using its type library and the **References** dialog box.

Advantages of Setting a Reference to a Type Library

Setting a reference to a type library for a component offers three advantages to client developers:

® Early binding

Setting a reference to a type library during design time enables early binding. Early binding enables client developers to find out about the methods and properties available from a component at compile time, rather than at run time.

When early binding is used, calls to a component are very efficient because they can use **vtable** binding.

® Syntax checking

The syntax of calls to a component is checked during development.

® Access to Help

Client developers have access to any context-sensitive Help that you have provided.

Article ID: Q159976
Creation Date: 25-NOV-1996
Revision Date: 17-JAN-1997

The information in this article applies to:

® Microsoft Active Server Pages, version 1.0

Summary

When you connect to Microsoft SQL Server from the Active Server Pages (ASP) ADO via named pipes, the client computer must be validated by SQL Server to use the named pipe. If SQL Server is on a different physical computer than the ASP files, the Active Server Pages might fail.

More Information

By default, Internet Information Server (IIS) attempts to connect to the SQL Server anonymously. To do this, it uses the Anonymous Logon information provided in Internet Service Manager (Inetmgr.exe). Typically this will be in the form of IUSR_MachineName where MachineName is the name of the server hosting IIS.

This account must be verified by the server hosting Microsoft SQL Server in order for the connection to occur properly. If this account is not verified, then the connection fails and provides only the line number in the script file of the attempted connection opening.

To resolve this problem, you can do one of the following:

1. Use the account information from Internet Service Manager to set up a local user account on the Windows NT Server that hosts SQL Server.
2. Make the Windows NT ID IUSR_MachineName a member of the domain where SQL server resides.
3. Enable the Windows NT ID Guest.

References

For more information about this behavior refer to the following articles in the Microsoft Knowledge Base:

Article-ID: **Q152828**
TITLE: IIS Queries to SQL Server Generate Error 1326

Article-ID: **Q142868**
TITLE: Authentication and Security Features

KBCategory: kbnetwork kbsetup kbhowto
KBSubcategory: AXSFDataBase AXSFCompADO
Additional reference words: 1.00 kbdsi

Article ID: Q159402
Creation Date: 13-NOV-1996
Revision Date: 17-JAN-1997

The information in this article applies to:

® Microsoft Active Server Pages, version 1.0 on the following platforms: Alpha, MIPS, Power PC, x86

Summary

When trying to use the Response.Redirect method in a server-side script,the following error can occur when the page is accessed:

Response
Headers already written to client.

More Information

The Redirect method of the Response object operates by sending a header to the client. This header causes the client to look to another URL location specified in the header. Because a header must come at the beginning of a document, it is not possible to place the Redirect method in a document with HTML code preceding it.

To work around this behavior you can use the buffering capabilities of the Response object. In doing this you can output HTML code into the buffer until you reach a point where you use the Redirect method. If at this point you need to redirect to another page, you clear the buffer and then issue the Response.Redirect.

The following example Active Server Pages (ASP) code demonstrates this concept:

```
<%
   ' Begin buffering the HTML
   ' Note this MUST happen before the initial <HTML> tag.
   Response.Buffer = True
   %>
   <HTML>
   <BODY>
   HTML code before potential redirect.<P>

   <%
   ' Change the following line as appropriate for your script
   If 1 = 1 Then
      Response.Clear
      Response.Redirect "filename.asp"
   End If
   %>
```

Use the following additional HTML code after the redirect:

```
<%
   ' The following causes the HTML to actually be sent to the client.
   ' Up to this point, no HTML has actually been downloaded to the client
   ' browser.
   Response.End
   %>
   </BODY>
   </HTML>
```

The above example always redirects to the file named Filename.asp.

KBCategory: kbcode kbprb kbhowto
KBSubcategory: AXSFHTML
Additional reference words: 1.00 kbdsi

Article ID: Q160682
Creation Date: 17-DEC-1996
Revision Date: 08-JAN-1997

The information in this article applies to:

® Microsoft Merchant Server, version 1.0

Summary

This article contains the answers to frequently asked questions about Microsoft Merchant Server.

More Information

What databases can be used with Merchant Server?

Any ODBC 2.5 compliant relational database management system (RDBMS) can be used with Merchant Server. Also, in the order pipeline you can write your own component(s) to get data from a legacy system.

In a multi-computer installation, what do I need on each computer?

You must have the controller and a router on the same computer. You can then install additional routers and store servers on multiple computers, depending on your scalability needs. Note that the router must be installed on a computer that has Internet Information Server (IIS). The relational database can be on any computer. Please refer to your Merchant Server documentation for more information.

Does Merchant Server 1.0 work with Active Server Pages?

No, it does not. However, this feature is being considered for future releases of Merchant Server.

What databases do the starter stores work with?

The Merchant Server starter stores work with SQL Server and Oracle databases.

For SQL Server, Merchant Server can run starter store SQL scripts during its setup. In the case of Oracle, Merchant Server installs Oracle SQL script files, and you need to manually run these scripts using a tool such as SQL*Plus. You can modify these SQL scripts and run them against any relational database that has an ODBC 2.5 compliant driver.

Why doesn't my store work with the America Online browser?

Testing has shown that the America Online browser cannot handle a URL with more than 72 characters. It is relatively easy to exceed 72 characters with Merchant Server. The America Online browser also has problems with semicolons (;) in the URL. This is an issue with America Online that will be resolved once America Online converts to newer browsers.

Why can't I access my stores as the administrator?

Make sure that you are typing the user ID (UID) and password corresponding to the admin_name and admin_password entries in the system registry under the following key:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\MerchantServer\Stores\
    <Store Name>\Admin
```

Also, if you have Basic (clear text) authentication enabled in IIS, you need to create a Windows NT user account corresponding to the administrator account for each store. If you don t need Basic authentication, simply clear the option in Internet Service Manager. You should also check the admin_ip entry in the registry, to see if the administrator access is restricted by the IP address.

The store will not start, and the error message is not descriptive. Why? And how can I get a more descriptive error message?

Try to connect to IIS (http://<machine name>), and see if you call browse your site. If you cannot connect, then the problem you are experiencing is not due to Merchant Server. You need to ensure that IIS is running properly.

To make sure that both the store service and router are running, do the following:

1. Launch the Merchant Server Administrator from Control Panel.
2. Click the Store Service tab, to see if the store service is running. Verify that the secure and insecure process count is set to 1 or more.
3. Click the Router tab, to ensure that it is running.
4. Click the Environment tab, and verify that all your stores are listed in the Installed Stores box.

Something is causing Merchant Server to stop or not to start at all. Use the Windows NT Event Viewer application to find out why the store service stopped or did not start. Whenever the store server encounters an error and exits, it reports the error in the Windows NT event log. Multiple messages are often logged, although usually only one message describes the root cause of the problem. Depending on the messages you see in the Windows NT event log, check all the registry entries and your database connectivity. Also, depending on the number of secure and insecure processes configured and the processor speed, Merchant Server store processes may take some time to start up, and you may occasionally see messages in the event log stating that a pipe could not be opened. In this case, it is a good idea to start the store service with only one insecure process, or wait for a while before browsing your stores. However, there may be other cases, like a store service shutting down, that will report pipe errors in the event log. In this case, you should investigate other errors reported in the event log.

Also, pay attention to the URL you typed. Note that Merchant Server URLs are case-sensitive. You should use all lowercase letters in the URL, except for the shopper ID. The shopper ID should be 0L for a new shopper, and 1L for the store administrator.

To have more descriptive error messages reported to your browser, you can also enable the Debug All Stores flag on the Environment tab of the Merchant Server administrator.

Can multiple stores use the same shopper table (shopper login to a mall)?

Yes. Use the same table name for the db_table_shopper entry for all your stores in the following registry entry:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\MerchantServer\Stores\
    <Store Name>\Database
```

I copied a store using Merchant Administrator, and I am now getting the error "Cannot initialize OrderEngine because merchant: Bad currency in registry." Why?

You receive this error if you copied a store that is configured to use the vPOS component for the payment stage, but you have not run vPOS setup to configure the newly copied store.

How do I use a Java applet running on a store?

The following is an example that was in the Adventure Works product.html:

```
<APPLET CODE="ThreeDObject.class" CODEBASE="[img]netvr/"
WIDTH=320 HEIGHT=240>
<PARAM NAME=imgDirectory VALUE="[img]netvr/rock/">
<PARAM NAME=filebase VALUE="rock">
<IMG SRC="[img]netvr/rock/rock.gif"><P>
</APPLET>
```

Is it possible to have guest and member shoppers?

Yes. Use the shopper.guest and shopper.lookup actions respectively. You can also use the shopper.new action to insert more information into the shopper table for a non-member shopper.

When using SSL, how can the warning dialog boxes be suppressed on an order.purchase action? Is it possible to secure the entire site?

If all the browsers recommended and tested by your store support redirection, specify "redirect=0" in the order.purchase action. Even though [sxform] or [sxurl] can be used with every Merchant Server action, you should typically use them for the order.purchase action. Because Merchant Server does not support enabling SSL on all of its virtual directories, it is not possible to secure the entire site. However, you should not be alarmed by this, as you can always use [sxform] or [sxurl] directives to run actions like order.purchase that contain confidential information. Also, there is no significant reason to secure things like product images and assets, because they can be saved locally and redistributed by a client.

Can I chain actions? Can I add multiple items to the basket without leaving the product page?

Microsoft Merchant Server 1.0 does not directly support chaining actions. There is no Merchant Server syntax that allows more than one action to be invoked. However, you can accomplish this by targeting a hidden frame with one action, and then calling the OnLoad event to run another action in a VBScript or a Javascript function.

To add multiple items to your basket without leaving the product page, use the method described in the preceding paragraph, but do nothing in the hidden frame. You would typically do this to display a client-side basket on the product page that also displays multiple products to the shopper. When a shopper clicks on an item, you call the order.additem action to add the item to the server-side basket, and also add the item to a client-side basket (typically an ActiveX control or a Java applet). As the target for order.additem is a hidden frame, the shopper can continue to add items without ever leaving that page, until it is time to check out.

How do I target a frame using the [xlink] directive?

You can use [xurl], which is very similar to [xlink] but does not place the URL into an <HREF> tag. You need to use the TARGET property for the <A> tag, and specify the desired frame. The following example illustrates the usage of [xurl]:

```
<A HREF="[xurl shopper.new error="error.html"]" TARGET="FrameName">
</A>
```

The optional components provided for certain stages in the order pipeline do not fit my model. What should I do?

Merchant Server provides API functions so you can write your own components and plug them into any stage in the order processing pipeline. You can also plug-in any third-party components available for stages such as shipping, handling, tax, and payment. The following is a list of some useful third-party applications:

® Taxware (for tax component)

® Tandata (for shipping and handling components)

® Verifone (for payment component)

Does Merchant Server use Secure Electronic Transactions (SET)?

Merchant Server ships with an evaluation version of vPOS (a payment component from Verifone), which uses its own custom implementation of SET between Merchant Server and the bank involved. This is because SET standards are still emerging. Currently, SSL is used to protect credit card information submitted by a shopper to the merchant.

Does Merchant Server support fractional quantities?

Merchant Server 1.0 does not support fractional quantities (for example, 0.5 pounds of cheese). The system only works with integral money and integral quantities, but "1" may mean 1 quarter of a pound or 1 hundredth gram.

Can Merchant Server Directives, value references and actions be used with ActiveX controls, VBScript and Javascript?

Yes. For some examples, see the Adventure Works starter store.

When I use Merchant Server Directives, I cannot retrieve more than 200 rows from the database. How can I increase this limit?

You can use the Merchant Administrator Control Panel to increase the number of rows returned. To do this, use the following steps:

1. Click the Environment tab of the Merchant Administrator.
2. Select an appropriate store environment.
3. Specify the maximum number of rows that any of your queries may return in the 'Maximum Query Size' box.

The default limit is set to 200 rows to prevent the accidental retrieval of huge result sets.

KBCategory: kbref kbfaq
KBSubcategory: MSrvGen
Additional reference words: 1.00 FAQ

Article ID: Q158737
Creation Date: 04-NOV-1996
Revision Date: 17-JAN-1997
The information in this article applies to:

® Microsoft Active Server Pages, version 1.0

Summary

This article provides information on how to create a simple query form using Active Server Pages and ActiveX Layout (ALX) files. This example displays a record from a database in the ALX, and then allows the user to query for a different record, which appears in the same form.

More Information

Below are two sections of code: a main page (Page1.asp) and the ActiveX Layout file Query.asp. Save these files to the same directory with the file names as given. Make sure that the directory the files are saved to has Execute and Read permissions in Internet Information Server (IIS) Manager.

Main Page: Page1.asp

```
<% Response.Expires = 0 %><html><head><title>Title</title></head>
<body><%If Not IsObject(Session("cn")) Then
  REM Create connection object if necessary
  Set cn = Server.CreateObject("ADODB.Connection")
  cn.Open "AdvWorks"

  REM Gather initial data
  Set rs = cn.Execute("SELECT * FROM CUSTOMERS WHERE CustomerID=1")
  Set Session("cn") = cn

  REM Place the information to be displayed in the ALX in session
  REM variables, because the ALX will be parsed AFTER this page loads.
  Session("ID") = rs(0)
  Session("Name") = rs(1)

Else
  On Error Resume Next

  REM Restore existing connection
  Set cn = Session("cn")

  REM Query is based on ID submitted in form
  MySQL = "SELECT * FROM CUSTOMERS WHERE CustomerID=" & Request.Form("ID")
  Set rs = cn.Execute(MySQL)
  If Err = 0 Then
    Session("ID") = rs(0)
    Session("Name") = rs(1)
  Else
    REM Query Failed
    Session("ID") = "None Found"
    Session("Name") = "None Found"
  End If

End If %><form name="HiddenForm" action="page1.asp" method="Post">
  <input name="ID" type="Hidden" maxlength="5" size="5" value="Nothing">

</form><script language="VBScript"><!--Sub SubmitForm
```

```
  Document.HiddenForm.ID.Value = Window.Html_Layout1.IDTextBox.Value
  HiddenForm.Submit

End Sub
-->
</script><objectclassid="CLSID:812AE312-8B8E-11CF-93C8-
00AA00C08FDF"id="Html_Layout1" style="LEFT:0;TOP:0"><param name="ALXPATH"
refvalue="query.asp"></object></body></html>
```

Layout File: Query.asp

```
<SCRIPT LANGUAGE="VBScript"><!--Sub CommandButton1_Click()
   Window.Parent.SubmitForm

End Sub--> </SCRIPT>
<DIV ID="Layout1" STYLE="LAYOUT:FIXED;WIDTH:225pt;HEIGHT:90pt;">
    <OBJECT ID="IDLabel"
     CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-00AA003F40D0"

STYLE="TOP:8pt;LEFT:8pt;WIDTH:99pt;HEIGHT:17pt;ZINDEX:0;">
        <PARAM NAME="Caption" VALUE="Customer ID">
        <PARAM NAME="Size" VALUE="3493;600">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
    </OBJECT>
    <OBJECT ID="IDTextBox"
     CLASSID="CLSID:8BD21D10-EC42-11CE-9E0D-00AA006002F3"

STYLE="TOP:25pt;LEFT:8pt;WIDTH:99pt;HEIGHT:16pt;TABINDEX:3;ZINDEX:4;">
        <PARAM NAME="VariousPropertyBits" VALUE="746604571">
        <PARAM NAME="Size" VALUE="3493;564">
        <PARAM NAME="Value" VALUE="<%=Session("ID")%>">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
    </OBJECT>
    <OBJECT ID="CommandButton1"
     CLASSID="CLSID:D7053240-CE69-11CD-A777-00DD01143C57"

STYLE="TOP:50pt;LEFT:8pt;WIDTH:99pt;HEIGHT:25pt;TABINDEX:2;ZINDEX:2;">
        <PARAM NAME="Caption" VALUE="Get Item">
        <PARAM NAME="Size" VALUE="3493;882">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="ParagraphAlign" VALUE="3">
    </OBJECT>
    <OBJECT ID="NameLabel"
     CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-00AA003F40D0"

STYLE="TOP:8pt;LEFT:116pt;WIDTH:99pt;HEIGHT:17pt;ZINDEX:3;">
        <PARAM NAME="Caption" VALUE="Company Name">
        <PARAM NAME="Size" VALUE="3493;600">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
    </OBJECT>
    <OBJECT ID="NameTextBox"
    CLASSID="CLSID:8BD21D10-EC42-11CE-9E0D-00AA006002F3"

STYLE="TOP:25pt;LEFT:116pt;WIDTH:99pt;HEIGHT:16pt;TABINDEX:1;ZINDEX:1;">
```

```
        <PARAM NAME="VariousPropertyBits" VALUE="746604571">
        <PARAM NAME="Size" VALUE="3493;564">
        <PARAM NAME="Value" VALUE="<%=Session("Name")%>">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
    </OBJECT>
</DIV>
```

When Page1.asp is requested from a browser. the following sequence of events occurs.

1. Browser requests Page1.asp from IIS.
2. IIS receives the request and recognizes the page as a script page.
3. IIS processes the script page.
4. In processing the script, a connection is stored as a session variable.
5. An initial query is performed, and two more session variables holding the CustomerID and the CustomerName are created.
6. IIS passes the resulting HTML code from Page1.asp back to the browser.
7. The browser begins displaying the page until it reaches the OBJECT tag that points to Query.asp.
8. The browser requests Query.asp from IIS.
9. IIS parses the Query.asp script tags and places the Name and ID into the text boxes in the layout.
10. IIS passes the Query.asp back to the browser, which interprets the file as a layout file (ALX).
11. The ALX appears, and the browser now waits for user input.
12. The user can input an ID into the first text box, and click the command button to perform the search.
13. The command button calls VBScript, which calls a procedure in Page1.asp.
14. The procedure in Page1.asp populates a hidden form and then submits the form to itself (Page1.asp).
15. Page1.asp is run as a script again.
16. Page1.asp recognizes that it has been called by a form and constructs a SQL query based on information passed.
17. It then populates the session variables ID and Name.
18. The page is passed back to the browser, which displays it, and again requests the Query.asp layout page.
19. Query.asp is parsed as a script and returns a valid ALX layout file.
20. The results appear on the screen.

Note This example requires the AdvWorks data source set up during theActive Server Pages installation.

References

For additional information, please see the following article in theMicrosoft Knowledge Base:

ARTICLE-ID: **Q157748**
TITLE: How to Modify .alx File Objects From Active Server Pages

KBCategory: kbprg kbhowto
KBSubcategory: AXSFHTML AXSFCompADO AXSFDataBase
Additional reference words: 1.00 kbdsi

What is Active Server Pages?

Active Server Pages is the server-side execution environment in IIS 3.0 that enables you to run ActiveX Scripts and ActiveX Server Components on the server. By combining scripts and components, organizations can create dynamic content and powerful web-based applications easily.

What is dynamic content?

Web pages that are customized for each user on the fly, based upon their actions or requests. For example, new visitors to your site can be shown a different welcome than returning users, or pages in an online catalog can be queries to a database so customers always see the most current information and availability.

Who should use Active Server Pages?

Organizations will use the Active Server Pages technology to put a web front-end on existing business solutions, or to create entirely new web-based applications. Since Active Server Pages provides a very open development environment, with support for both VBScript and JScript, organizations can leverage the investments they already have in these scripting languages.

Who can create dynamic content with Active Server Pages?

Webmasters, IS professionals, and programmers familiar with HTML and languages such as Visual Basic, JavaScript, Perl, REXX, or C++.

What do I need to run Active Server Pages?

Active Server Pages requires Internet Information Server 2.0 and Windows NT Server 4.0 or Windows NT Workstation 4.0 and Peer Web Services.

How much will Active Server Pages cost?

Active Server Pages is a component of IIS 3.0, which is free, downloadable, and integrated feature of Windows NT Server 4.0.

What can Active Server Pages do for my business?

Develop a new generation of Web-based applications, including extending sales and customer service to the Web, and providing access to corporate databases and applications to any browser on an intranet.

What ActiveX Server Components are supported?

Active Server Pages allows organizations to extend the power of scripting on the server with ActiveX Server Components. These components can be created using VB, Java, VC++, etc.

What scripting languages does Active Server Pages support?

Active Server Pages provides native support for both JScript and VBScript. ActiveX scripting plug-ins are available for REXX, PERL, and Python.

What browsers does Active Server Pages support?

Active Server Pages can work with any Web browser. The output of an ASP file is plain HTML, the content of which can be customizes for the capabilities of the client.

Does Active Server Pages maintain state for me?

Yes. Active Server Pages allows you to define application and session variables that can be carried across multiple pages in a Web site. This can be as simple as remembering a user's name, and is necessary in applications such as online shopping to track product selections.

What about legacy data?

Active Server Pages makes it easy to bring legacy database applications to the Web.

On which platforms does Active Server Pages run?

Active Server Pages will run on NT 4.0 Server, NT 4.0 Workstation with Peer Web Services, and Windows 95 with Personal Web Server. NT 3.51 is NOT supported. Windows NT 4.0 on MIPS is also not supported by Active Server Pages.

Is Active Server Pages secure?

Yes. Active Server Pages is a component of Internet Information Server, and thus uses Windows NT Security. ASP files can be easily restricted to just certain users through secure Windows NT authentication, basic Web authentication, or client-side certificates. For additional security, all client to server communications can be secured with SSL.

What data sources can my Web application integrate with?

An Active Server Pages application can integrate with any ODBC-compliant databases including Microsoft SQL Server, Oracle, Sybase, Informix, and DB2 databases. Any OLE 2 application, such as Lotus Notes or Microsoft Excel, can also be scripted to access or process information. You can also write components to access online data feeds and legacy mainframes.

How does Active Server Pages compare to CGI?

Active Server Pages provides all of the functionality of CGI applications in an easier-to-use and more robust environment.

Active Server Pages is an easier way for your server to access information in a form not readable by the client (such as an SQL database) and then act as a gateway between the two to produce information that the client can view and use.

With CGI, the server creates as many processes as the number of client requests received. The more concurrent requests there are, the more concurrent processes created by the server. However, creating a process for every request is time-consuming and requires large amounts of server RAM. In addition, this can restrict the resources available for sharing from the server application itself, slowing down performance, and increasing wait times on the Web.

Active Server Pages instead runs in the same process as the Web Server, more handling client requests faster and more efficiently. It is much easier to develop dynamic content and Web applications with Active Server Pages.

How does Active Server Pages compare to ISAPI applications?

ISAPI applications require all of the programming and layout to be contained in a dll file written in C++. ISAPI applications are thus more difficult to create and maintain. With ASP files, an HTML writer can script an external component and format the output. Active Server Pages separates the layout and design from the business logic.

How does Active Server Pages compare to Perl?

Perl and other scripting languages are not robust development tools by themselves. Active Server Pages provides a familiar framework and objects for building complex applications that require data from relational databases and legacy sources.

Active Server Pages supports virtually any scripting language to build these applications. Third parties are currently developing additional scripting engines, such as Perl, which we will announce when they are ready.

Can Active Server Pages use Java?

Yes. Active Server Pages supports ActiveX server components written in any language, including Java. In addition, Active Server Pages includes Microsoft's Windows reference standard Java Virtual Machine.

Can I use existing Automation servers?

There really are no minimum requirements for a component, beyond supporting Automation through IDispatch.

You can optionally provide the following two methods on IDispatch:

1. OnStartPage
2. OnEndPage.

Refer to "Creating ASP Components" in the Programmer's Reference for more details.

What does Active Server Pages do better than other Web application tools?

Leveraging your existing skills and knowledge, data sources, components, and applications to quickly bring them to the Web. Other tools create either static HTML, or lock you into a non-standard programming model or language. Active Server Pages is based upon the leading industry standards, making it easy to build, maintain, and evolve powerful interactive Web applications.

How does Active Server Pages compare to Netscape LiveWire?

Netscape LiveWire requires the use of JavaScript, while Active Server Pages supports the use of virtually an scripting language, with native support for VBScript and Jscript. Active Server Pages supports components written in any language while LiveWire supports only Java components.

LiveWire applications must be manually compiled after each change, and then the application stopped and restarted. Active Server Pages recognizes when an ASP file changes, and automatically recompiles the application at the next request.

```
'when a session starts, redirect the user to a logon page
Sub Session_OnStart
    If Session("username") = "" Then
        'save the name of the page the user wanted to visit
        Session("startpage") = Request.ServerVariables("SCRIPT_NAME")
        'redirect them to a logon page
        Response.Redirect "profile.htm"
    End If
End Sub
```

```
'when a session starts, save data connection information
Sub Session_OnStart
    '==Visual InterDev Generated - DataConnection startspan==
    '--Project Data Connection
    Session("StateU_ConnectionString") = _
        "DSN=StateU;UID=sa;PWD=;" & _
        "APP=Microsoft (R) Developer Studio;" & _
        "WSID=SERVERNAME;DATABASE=StateU"
    Session("StateU_ConnectionTimeout") = 15
    Session("StateU_CommandTimeout") = 30
    Session("StateU_RuntimeUserName") = "sa"
    Session("StateU_RuntimePassword") = ""
    '==Visual InterDev Generated - DataConnection endspan==
End Sub
```

Click here to connect to the Microsoft Internet Development Web site.
{ewc mvimg, mvimage,!intjump.bmp}

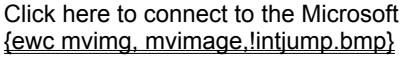
This site covers development of ActiveX controls, use of ActiveX controls in Web pages and in other applications, development of ActiveX document servers, development of applications that use Internet communications technology, and development of server-side functionality, including database access, ISAPI filters, applications, and (soon) server-side scripting and controls.

Click here to connect to the Microsoft Internet Information Server Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The Microsoft Internet Information Server, the only web server integrated into Windows NT Server, is the most powerful platform for a new generations of Web applications.

Click here to connect to the Microsoft Exchange Server Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The Microsoft Exchange Server Web site is the premier source of information about Microsoft Exchange Server. On this site, you will find the latest news, product information, and technical information about the only server to embrace Internet standards and extend rich messaging and communication to businesses of all sizes-Microsoft Exchange Server.

Click here to connect to the Microsoft SQL Server Web site.

{ewc mvimg, mvimage,!intjump.bmp}

Visit this site to learn about Microsoft SQL Server, a scalable, high-performance database management system designed specifically for distributed client-server computing.

Click here to connect to the Microsoft Windows Family Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Visit this Web site to learn more about the Microsoft family of Windows products, including Windows 95, Windows NT Workstation, Windows NT Server, and Windows CE.

Click here to connect to the Microsoft Windows NT Server Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Find out all you need to know about Windows NT Server by visiting this site.

Click here to visit the Microsoft Interactive Media Technology Web site.
{ewc mvimg, mvimage,!intjump.bmp}

This is a great site for authors and developers to discover about what's new in interactive media technology, review press information, take a tour, find out about compatible products, and download the latest software.

Click here to connect to the Microsoft Press Releases Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The site for Microsoft News and Corporate Information.

Click here to connect to the Microsoft BackOffice Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Visit this site to learn more about Microsoft BackOffice, an integrated family of server products that provides the platform for a new generation of business applications. Members of the BackOffice family include:

® Microsoft Exchange Server

® Microsoft SQL Server

® Microsoft Proxy Server

® Microsoft Systems Management Server

® Microsoft SNA Server

® Microsoft Merchant Server

® Microsoft Transaction Server

® Microsoft Commercial Internet System

® Microsoft BackOffice 2.5

Click here to connect to the Microsoft ActiveX SDK for Macintosh Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Visit this site to learn about Macintosh ActiveX, Microsoft's native Macintosh implementation of ActiveX controls.

Article ID: Q163159
Creation Date: 04-FEB-1997
Revision Date: 06-FEB-1997

The information in this article applies to:

® Microsoft Access 97

Summary

Advanced: Requires expert coding, interoperability, and multiuser skills.

Warning Although this article discusses Microsoft Access security features, any information you send over the Internet with the techniques described in this article is sent unencrypted. To send encrypted information over the Internet, you must use a protocol that sends client certificates, such as Secure Sockets Layer (SSL). Note that client certificates cannot be used on Personal Web Server for Windows 95. ANY USE BY YOU OF THE METHODS PROVIDED IN THIS ARTICLE IS AT YOUR OWN RISK. Microsoft provides this sample "as is" without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability and/or fitness for a particular purpose.

This article describes a technique you can use to create Active Server Pages (ASP) files that allow you to type a username and password in an HTML form in order to query a secure Microsoft Access database. This article describes how to use the technique when exporting a query and when exporting a form.

More Information

There are three main steps to using ASP files to query a secure Microsoft Access database:

® Use ODBC Administrator to create a System DSN that points to the workgroup information file (System.mdw) you use with your secured database.

® Create an HTML form that requests a username and password. The HTML form passes the values to parameters in your ASP file.

® Modify the script in the ASP file to use the HTML form to request the username and password parameters in order to authenticate user access to your database.

You must use an HTML form to enter the username and password, and then pass that information to the ASP files. You cannot configure Microsoft Internet Information Server (IIS) Basic Authentication or NT Challenge/Response to achieve this functionality because those IIS options authenticate users against Microsoft Windows NT permissions, not Microsoft Access security accounts. It is possible to use Basic Authentication and NT Challenge/Response with Microsoft SQL Server databases because SQL Server can be integrated with NT Security. Microsoft Access security does not provide that capability.

Note The following example assumes that you are familiar with Microsoft Access security and will require you to create a secured copy of the Northwind sample database shipped with Microsoft Access.

This example contains the following sections:

® Creating a System DSN for a Secure Microsoft Access Database

® Creating the ASP Files and the HTML Logon Form

® Customizing the ASP Files and HTML Logon Form

® Testing the Query

Creating a System DSN for a Secure Microsoft Access Database

1. Double-click the ODBC icon in Control Panel on your Web Server.
2. In the ODBC Data Source Administrator dialog box, click the System DSN tab.

3. Click the Add button.
4. Select Microsoft Access Driver, and then click Finish.

Note If the Microsoft Access Driver does not appear, it is not installed on your Web server. For information about installing the driver on your Web server, search the Help Index for "Microsoft Access Desktop driver," or ask the Microsoft Access 97 Office Assistant.

---

5. In the ODBC Microsoft Access 97 Setup dialog box, type NorthwindASP in the Data Source Name box.
6. Click the Select button and browse to select Northwind2.mdb. Click OK.
7. In the System Database box, click Database, and then click the System Database button. Browse to select Northwind2.mdw, and then click OK.
8. Note that you have the option to click the Advanced button in the ODBC Microsoft Access 97 Setup dialog box, and set a default Login name and Password for the System DSN. Any of your ASP files that do not provide a username and password will use the default settings.
9. Click OK to close the ODBC Microsoft Access 97 Setup dialog box.
10. Click OK to close the ODBC Data Source Administrator dialog box.

Creating the ASP Files and the HTML Logon Form

In this section, you create a query with username and password parameters, and then export the query to ASP format. When you create ASP files from a parameter query, Microsoft Access automatically creates an HTML form for entering the parameters. This is an easy way to create the HTML form you need to collect the username and password information. However, you do not have to use this technique to create the HTML form; you can use Notepad or another tool, such as Microsoft Front Page 97, to create your own HTML Logon form.

1. Start Microsoft Access.
2. Open Northwind2.mdb.
3. Create the following new query called GetUserPass based on the Customers table:

Query	**GetUserPass**
Type:	Select Query
Field:	CustomerID
Table:	Customers

4. On the Query menu, click Parameters.
5. Type the following in the Query Parameters dialog box, and then click OK.

Parameter	**Data Type**
[UserParam]	Text
[PassParam]	Text

6. Save the GetUserPass query and close it.
7. On the File menu, click Save as HTML. The "Publish to the Web Wizard" dialog box opens. Click Next.
8. Click the Queries tab, and select the GetUserPass query. Click the Forms tab, and select the Customers form. Click Next, and Next again.
9. You will be asked to select a format type to create. Select Dynamic ASP (Microsoft Active Server Pages). Click Next.
10. In the Data Source Name box, enter NorthwindASP, and click Next.
11. The Publish Objects Locally button should be selected. Note the folder where the ASP files will be exported to. You do not need to make any more selections, so you can click Finish at this point.
12. Click OK in each of the two Enter Parameter Value dialog boxes that appear.
13. The ASP output creates four files: GetUserPass_1.asp, GetUserPass_1.HTML, Customers_1.asp, and Customers_1alx.asp.

Customizing the ASP Files and the HTML form

Note This section contains information about editing ASP and HTML files, and assumes that you are familiar with editing HTML files, Active Server, and Visual Basic Scripting. Microsoft Technical Support engineers do not support customization of any HTML, HTX, IDC, or ASP files.

1. Use Notepad or another text editor to open the GetUserPass_1.asp file. At the top of the file, you will see the following code:

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>GetUserPass</TITLE>
</HEAD>
<BODY>
<%
Param = Request.QueryString("Param")
Data = Request.QueryString("Data")
%>
<%
If IsObject(Session("NorthwindASP_conn")) Then
   Set conn = Session("NorthwindASP_conn")
Else
   Set conn = Server.CreateObject("ADODB.Connection")
   conn.open "NorthwindASP","",""
   Set Session("NorthwindASP_conn") = conn
End If
%>
Replace the above code with the following (the rest of the code should
remain the same):
<%
If IsObject(Session("NorthwindASP_conn")) Then
   Set conn = Session("NorthwindASP_conn")
Else
   If Request.Form("[UserParam]") = "" then
    response.redirect "GetUserPass_1.HTML"
   Else
    Set conn = Server.CreateObject("ADODB.Connection")
    conn.open "NorthwindASP", Request.Form("[UserParam]"), _
      Request.Form("[PassParam]")
    Set Session("NorthwindASP_conn") = conn
   End If
End If
%>
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>GetUserPass</TITLE>
</HEAD>
<BODY>
```

2. Save the GetUserPass_1.asp file and close it.
3. Use Notepad or another text editor to open the Customers_1.asp file. At the top of the file you will see the following code:

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
```

```
<TITLE>Customers</TITLE>
</HEAD>
<BODY>
<%
If IsObject(Session("NorthwindASP_conn")) Then
   Set conn = Session("NorthwindASP_conn")
Else
   Set conn = Server.CreateObject("ADODB.Connection")
   conn.open "NorthwindASP","",""
   Set Session("NorthwindASP_conn") = conn
End If
%>
```

Replace the above code with the following (the rest of the code should remain the same):

```
<%
If IsObject(Session("NorthwindASP_conn")) Then
   Set conn = Session("NorthwindASP_conn")
Else
   If Request.Form("[UserParam]") = "" then
     response.redirect "GetUserPass_2.HTML"
   Else
     Set conn = Server.CreateObject("ADODB.Connection")
     conn.open "NorthwindASP", Request.Form("[UserParam]"), _
       Request.Form("[PassParam]")
     Set Session("NorthwindASP_conn") = conn
   End If
End If
%>
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>Customers</TITLE>
</HEAD>
<BODY>
```

Save the Customers_1.asp file and close it. These changes will make the ASP file use the HTML form to request a username and password if the user had not yet entered a username and password for the session.

4. Use Notepad or another text editor to open the GetUserPass_1.HTML file. By default, the HTML form uses the GET method to submit its data. Get variables appear in the address box of Web browsers. Therefore, you must change the GET method to the POST method if you do not want your password to be visible in the address box of your Web browser. Locate the following line in the GetUserPass_1.HTML file:

   ```
   <FORM METHOD="GET" ACTION="GetUserPass_1.asp">
   and change it to:
   <FORM METHOD="POST" ACTION="GetUserPass_1.asp">
   ```

5. Text boxes use an Input Type setting of Text by default. In order to prevent your password from being visible in the text box on your form, you must change the Input Type to Password. Locate the following line in the GetUserPass_1.HTML file:

   ```
   [PassParam] <INPUT TYPE="Text" NAME="[PassParam]"><P>
   and change it to:
   [PassParam] <INPUT TYPE="Password" NAME="[PassParam]"><P>
   ```

6. In Notepad, on the File menu, click Save. You will need to create another HTML form so do not close Notepad.
7. In Notepad, on the File menu, click Save As.
8. In the File Name box, type GetUserPass_2.HTML.

9. In the Save As Type box, select All Files, and click Save.
10. Locate the following line in the GetUserPass_2.HTML file

    ```
    <FORM METHOD="POST" ACTION="GetUserPass_1.asp">
    ```

    and change it to:

    ```
    <FORM METHOD="POST" ACTION="Customers_1.asp">
    ```

11. On the File menu, click Save. On the File menu, click Exit.
12. Copy GetUserPass_1.HTML, GetUserPass_2.HTML, GetUserPass_1.asp, Customers_1.asp, and Customers_1alx.asp to a folder on your Web Server computer where you have both Read and Execute permission.

    For more information about configuring Microsoft Internet Information Server (IIS) permissions, please refer to the IIS Help Index, and see the following article in the Microsoft Knowledge Base:

    ARTICLE-ID: **Q160754**
    TITLE: Error "HTTP/1.0 403 Access Forbidden" Browsing IDC Page

Testing the Query

1. Start Microsoft Internet Explorer 3.0, or another Web browser program.
2. Type the Uniform Resource Locator (URL) in the address box of your Web browser to view GetUserPass_1.asp. For example, if you saved your ASP files in a folder called Test in the wwwroot folder of your Web Server, type:

    http://<servername>/test/GetUserPass_1.asp

    Note that the URL depends upon where your files are located on the Web Server and that Internet Explorer 3.0 with the HTML Layout Control is necessary to view forms exported to ASP.
3. The GetUserPass_1.HTML form opens in your Web browser with a [UserParam] box, a [PassParam] box, and a Run Query button. Type Admin in both boxes, and then click the Run Query button. The GetUserPass_1.asp file opens and displays a list of CustomerIDs.
4. Type the appropriate URL to view Customers.asp and the Customers form will open. You will not be prompted for a username and password because the Session began when you first logged on. The Session ends when the Web browser is closed. A Session will also end if a user does not request or refresh a page within the timeout period. The Session Timeout property default is 20 minutes and can be changed in your ASP scripts. You can also end the Session by using the Abandon method of the Session object in your scripts. For more information about using the Abandon method and setting the Timeout property for the Session object, please refer to your online ASP documentation.

Note If you type an incorrect username or password, you receive the following error:

Microsoft OLE DB Provider for ODBC Drivers error '80004005'
Not a valid account name or password.

References

For more information about how to create and modify ASP files, please refer to your Microsoft ASP online documentation.

For more information about IIS authentication, security, and Secure Sockets Layer (SSL), please refer to your IIS online documentation, or see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q142868**
TITLE: Authentication and Security Features

For more information about Microsoft Access security, search the Help Index for "security, overview," or ask the Microsoft Access 97 Office Assistant.

```
<% If Not (IsEmpty(Request("txtID"))) Then
    'request came from Submit button on form
    username = Request("txtName")
    id= Request("txtID")
    major = Request("Major")
%>
Welcome, <%=username%>, to State University.
<P>I see you are interested in <%=major%>.
<% End If %>
<P>
<FORM NAME=frmProfile ACTION=profile.asp METHOD=Post>
...
```

Article ID: Q163014
Creation Date: 31-JAN-1997
Revision Date: 06-FEB-1997

The information in this article applies to:

® Microsoft Access 97

Symptoms

Advanced: Requires expert coding, interoperability, and multiuser skills.

When you browse to an ASP file that was exported with Microsoft Access 97, the Format property of fields in Microsoft Access tables or queries is not preserved.

Cause

When tables or queries are exported to ASP format, Microsoft Access 97 generates an ASP file that retrieves and displays the data from the record source, but the export functionality is not designed to preserve format properties. Numbers will be displayed in General Number format, Dates will be displayed as General Date, Times will be displayed in Long Time format, and Yes/No fields will be displayed in True/False format.

Resolution

There are two possible workarounds for preserving formats:

Method 1: Using the Format Function in a Query

This example uses the sample database Northwind.mdb.

1. Create the following query based on the Order Details table. Name it qryFormatTest. For each of the formatted field(s) you are trying to export, you must create a calculated field in the query using the Format() function as indicated below.

Query	**qryFormatTest**
Field:	OrderID [Criteria: <10300]
Field:	ProductID
Field:	NewPrice: Format ([Unit price], "Currency")
Field:	Quantity
Field:	NewDiscount: Format ([Discount], "0%")

2. Save and close the query. Select the query in the Database window.
3. On the File menu, click Save As/Export.
4. In the Save As dialog box, click to select "To an External File or Database," and click OK. Note that the "Save Query 'qryFormatTest' In" dialog box appears.
5. In the Save As Type box, select Microsoft Active Server Pages (*.asp) and type qryFormatTest.asp in the File Name box. Note the folder where the files will be exported to. Click Export. Note that the Microsoft Active Server Pages Output Options dialog box appears.
6. In the Data Source Name box, enter the name of a System DSN that points to the sample database Northwind.mdb.

   For more information about how to define a system DSN, search the Help index for "ODBC, setting up data sources," and see the following article in the Microsoft Knowledge Base:

   ARTICLE-ID: **Q159682**
   TITLE: "Data Source Name Not Found" Err Msg Opening Web Page
7. In the Server URL box, enter the URL that points to the Web Server location where your ASP files will be stored. For example, if you store the ASP files in the \ASPsamp folder on the \\PubTest server, type "http://pubtest/aspsamp/" (without the quotation marks) as your Server URL. Click OK. The ASP output

creates the file qryFormatTest.asp.

8. Copy qryFormatTest.asp to a folder on your Web Server computer where you have Execute permission. For more information about configuring Microsoft Internet Information Server (IIS) permissions, please refer to the IIS Help Index, and see the following article in the Microsoft Knowledge Base:

   ARTICLE-ID: **Q160754**
   TITLE: Error "HTTP/1.0 403 Access Forbidden" Browsing IDC Page

9. Start Microsoft Internet Explorer 3.0, or another Web browser program.
10. Type the Uniform Resource Locator (URL) in the address box of your Web browser to view qryFormatTest.ASP. For example, if you saved your ASP file in a folder called Test in the wwwroot folder of your Web Server, type:

    http://<servername>/test/qryFormatTest.ASP

    Note that the URL depends upon where your files are located on the Web Server.
11. Note that the NewPrice and NewDiscount fields have formatting applied.

Method 2: Modifying the ASP File Using VB Script

Note This section contains information about editing ASP files and assumes that you are familiar with editing HTML files, Active Server, and Visual Basic Scripting. Microsoft Access Technical Support engineers do not support customization of any HTML, HTX, IDC, or ASP files.

This example uses the sample database Northwind.mdb.

1. Create the following query based on the Order Details table. Name it qryFormatTest.

Query	**qryFormatTest**
Field:	OrderID [Criteria: <10300]
Field:	ProductID
Field:	UnitPrice
Field:	Quantity
Field:	Discount

2. Save and close the query. Select the query in the Database Window.
3. On the File menu, click Save As/Export.
4. In the Save As dialog box, click to select "To an External File or Database," and click OK. Note that the "Save Query 'qryFormatTest' In" dialog box appears.
5. In the Save As Type box, select Microsoft Active Server Pages (*.asp) and type qryFormatTest.asp in the File name box. Note the folder where the files will be exported to. Click Export. Note that the Microsoft Active Server Pages Output Options dialog box appears.
6. In the Data Source Name box, enter the name of a System DSN that points to the sample database Northwind.mdb.

   For more information about how to define a system DSN, search the Help Index for "ODBC, setting up data sources," and see the following article in the Microsoft Knowledge Base:

   ARTICLE-ID: **Q159682**
   TITLE: "Data Source Name Not Found" Err Msg Opening Web Page

7. In the Server URL box, enter the URL that points to the Web Server location where your ASP files will be stored. For example, if you store the ASP files in the \ASPsamp folder on the \\PubTest server, type "http://pubtest/aspsamp/" (without the quotation marks) as your Server URL. Click OK. The ASP output creates the file qryFormatTest.asp.
8. Use Notepad or another text editor to open the qryFormatTest.asp file. Towards the bottom of the file you will see the following code which is a combination of HTML and Active Server Scripting:

```
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial"
```

```
COLOR=#000000><%=Server.HTMLEncode(rs.Fields("UnitPrice").Value)%>
<BR></FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial"

COLOR=#000000><%=Server.HTMLEncode(rs.Fields("Quantity").Value)%>
<BR></FONT ></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial"

COLOR=#000000><%=Server.HTMLEncode(rs.Fields("Discount").Value)%>
<BR></FONT></TD>
```

To format the UnitPrice field as Currency and the Discount field as Percent, you must modify the code so it uses the VB Script FormatCurrency and FormatPercent functions:

```
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial"
COLOR=#000000><%=Server.HTMLEncode(formatcurrency(rs.Fields
"UnitPrice").Value))%><BR></FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial"

COLOR=#000000><%=Server.HTMLEncode(rs.Fields("Quantity").Value)%>
<BR></FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial"
COLOR=#000000><%=Server.HTMLEncode(formatpercent(rs.Fields
("Discount").Value,0))%><BR></FONT></TD>
```

Please refer to your VB Script Language Reference available in the ASP online documentation for more information about the VB Script Format functions.

9. Copy qryFormatTest.asp to a folder on your Web Server computer where you have Execute permission. For more information about configuring Microsoft Internet Information Server (IIS) permissions, please refer to the IIS Help Index. and see the following article in the Microsoft Knowledge Base:

   ARTICLE-ID: **Q162975**
   TITLE: Permissions Necessary to View HTML, IDC, and ASP Files
10. Start Microsoft Internet Explorer 3.0, or another Web browser program.
11. Type the Uniform Resource Locator (URL) in the address box of your Web browser to view qryFormatTest.ASP. For example, if you saved your ASP file in a folder called Test in the wwwroot folder of your Web Server, type:

    http://<servername>/test/qyrFormatTest.ASP

    Note the URL depends upon where your files are located on the Web Server.
12. Note that the UnitPrice and Discount fields have formatting applied.

More Information

Method 2 may be a better choice because the output will have right- justified Currency fields so the decimal point appears in the same position throughout the column. Method 1 will output the field as left-justified Text which may not line up the decimal point in the same position for each record. The disadvantage with Method 2 is that VB Script has the FormatCurrency, FormatNumber, FormatDateTime, and FormatPercent functions, but does not have a Format function where custom formats can be supplied.

References

For more information about exporting tables or queries to ASP, search the Help Index for "ASP files," and then "Export a datasheet to dynamic HTML format."

```
<!-- Page Header -->
<table ... >
<tr>
...
</tr>
<tr>
    <td bgcolor="#FFFFCC"> <font size="-1">  
    For Student ID <%=Session("id")%> </font>
    </td>
</tr>
</table>
<!-- end of page header -->
```

```
<%Session("username") = username
Session("id") = id
Session("major") = major
%>
```

A hit counter informs a user of how many users have visited the Web site. In this exercise, you will implement a hit counter for the State University Web site by using server-side script.

u Add a hit counter

1. Open the file global.asa in the Visual InterDev Source Editor.
2. Create an Application_OnStart event procedure, and add server-side script that creates an application variable named hitcounter and initializes it to zero.
3. In the Session_OnStart event procedure, before the script that redirects users to profile.asp, add server-side script that performs the following tasks:
   a. Locks the hit counter **Application** variable.
   b. Increments the hit counter **Application** variable by one.
   c. Unlocks the hit counter **Application** variable.

   For information about writing event procedures for the **Application** and **Session** objects, see Using Events in the Global.asa File.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}
4. Save your changes to global.asa.

u Test the hit counter

1. Open home.asp in the Visual InterDev Source Editor.
2. At the bottom of the page, before the </BODY> tag, add server-side script to output the value of the hit counter.
3. Save your changes to home.asp.
4. To test the hit counter **Application** variable, start a new session, and then go to home.asp.

u Save the hit counter to a file

The hit counter is saved in an **Application** object variable, so if the Web server is rebooted, the hit counter will be reset to zero.

To prevent the hit counter from resetting to zero each time the Web server shuts down and starts up again, use the **TextStream** component to save the hit counter information to a text file when you end the application.

1. In the global.asa file, create an Application_OnEnd event procedure.
   a. Create an instance of the **FileSystemObject** object by calling the **CreateObject** method of the **Server** object.
   b. Create a new text file by calling the **CreateTextFile** method of the **FileSystemObject** object. If there is an existing text file, create a new one.
   c. Write a line to the text file that contains the value of the hit counter **Application** variable.
   d. Close the file by calling the **Close** method.

   To see an illustration of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}

   For information about writing to a text file, see The File Access Component.
2. Save your changes to global.asa.
3. Test your changes by stopping the Web server and locating the new text file.

   To stop the Web server, run the Internet Service Manager, right-click the WWW service for your Web server, and then click **Stop**.

u Read the hit counter from a file

When you start the Web server, the Application_OnStart event procedure will reset the hit counter variable to zero, unless you read the new text file you created.

1. In global.asa, add script to the Application_OnStart event procedure.

{ewc mvimg, mvimage,!tip.bmp}

a. Create an instance of the **FileSystemObject** object by calling the **CreateObject** method.

b. Open the file containing the hit counter value by calling the **OpenTextFile** method of the **FileSystemObject** object.
{ewc mvimg, mvimage,!tip.bmp}

c. Read the value of the hit counter from the file, and write it to the hit counter **Application** object variable.

d. Close the file by calling the **Close** method.

To see an illustration of how your code should look, click this icon.
{ewc mvimg, mvimage,!code.bmp}

2. Save your changes to global.asa.
3. To test your changes, preview the page home.asp in the Visual InterDev InfoViewer.

   To make testing easier, you can edit the hit counter file manually, after stopping the Web server.

Note Each time you want to test the script in the Session_OnStart event procedure that increments the hit counter, you need to start a new Web session. To start a new session, start a new instance of Microsoft Internet Explorer.

Each time you want to test the script in the Application_OnEnd and Application_OnStart event procedures, you need to stop and restart the Web application. To stop and restart the application, you need to stop the WWW service of your Web server, and then restart it.

With Internet Information Server, stop the WWW service by running the Internet Service Manager, right-clicking the WWW service for your Web server, and then clicking **Stop**. To restart the WWW service, click **Start**.

Article ID: Q163485
Creation Date: 20-FEB-1997
Revision Date: 25-FEB-97

The information in this article applies to:

® Microsoft Internet Information Server, version 3.0

Symptoms

When you use Microsoft Internet Information Server, if you place a period (".") in a browser's command line after any script-mapped file name, you receive unexpected results. The browser produces a document that contains the scripting information as well as other data in the file.

For example, if you enter:

http://server_name/asp_directory/file.asp.

You receive something like:

```
<% emailx=request.form("email")
   remarkx=request.form("remark") Set Conn =
   Server.CreateObject("ADODB.Connection") Conn.Open "Local SQL
   Server", "sa", "DTide" Set RS = Conn.Execute("insert into
   Web_data.dbo.ASP_data(email,remark) values('" & emailx &
   "','" & remarkx & "')") %>

Your information has been added to our database.
```

The browser should return a confirmation web page, without the script.

Cause

The problem affects any script-mapped files requested from a virtual directory that has both read and execute permissions set. Adding one or more extra periods onto the end of the URL causes the file to be displayed in the browser, instead of run on the server. This allows end users to see information that may be confidential, such as server-side script logic (for example, the discount applied to the retail price from a database). This problem affects any file in the script-map list, such as .asp, .ht., .id, .PL, and so on.

This problem only occurs on virtual directories that have both read and execute access. If read is disabled, the server-side information is not viewable by the end user.

Resolution

To resolve this problem, do either of the following:

® Do not place script-mapped files in a directory that has both read and execute permissions.

-or-

® Get the hotfix mentioned below.

Note You can automatically download this hotfix from the Internet Information Server Fix topic in the Resources\Microsoft Tools section of the Library.

Status

Microsoft has confirmed this to be a problem in Microsoft Internet Information Server version 3.0. A supported fix is now available, but has not been fully regression-tested and should be applied only to systems experiencing this specific problem. Unless you are severely impacted by this specific problem, Microsoft recommends that you wait for the next Service Pack that contains this fix. Contact Microsoft Technical Support for more information.

This hotfix has been posted to the following Internet location. You can download any of these self-extracting files from the following service:

Internet (anonymous FTP) ftp ftp.microsoft.com Change to the Bussys/Winnt/Winnt-public/Fixes/Usa/NT40/ASP folder. Get Readme.txt (for instructions on downloading and installing the hotfix).

Contact Microsoft Technical Support for more information.

KBCategory: kbusage kbbug1.00 kbbug2.00 kbbug3.00
KBSubcategory: ntnetserv
Additional reference words: 1.00 2.00 3.00 prodnt 4.00

Article ID: Q159977
Creation Date: 25-NOV-1996
Revision Date: 17-JAN-1997
The information in this article applies to:

® Microsoft Active Server Pages, version 1.0

Summary

Microsoft Active Server Pages can be used to prevent Web browsers from calling a document that is supposed to be displayed as a part of a frameset.

More Information

Web documents that are part of a frameset are often not designed for individual display. To prevent users from viewing these documents outside of their appropriate frameset, you can use the Internet Information Server (IIS) Active Server Pages (ASP) "Response.Redirect" and "Request.ServerVariables" methods to redirect Hypertext Transfer Protocol (HTTP) requests to the frameset page.

Assuming the following document structure:

```
Frameset Page (mainfrm.htm)
      Frame 1 (frame1.asp)
      Frame 2 (frame2.asp)
```

place the following code in frame1.asp or frame2.asp before the opening<HTML> tag:

```
<%
   If (Request.ServerVariables("HTTP_REFERER") = "") Or _
      (Left(Request.ServerVariables("HTTP_REFERER"),42) <> _
      "http://www.myserver.com/AppDir/mainfrm.htm") Then
      Response.Redirect "http://www.myserver.com/AppDir/mainfrm.htm"
   End If
   %>
```

Note You must put this code at the beginning of your document because "Response.Redirect" does not work if any HTML code occurs before it. For more information on this, please see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q159402**
TITLE: How To Use Response.Redirect in a Server Script

In this code, the first part of the If statement (Request.ServerVariables("HTTP_REFERER")= "") checks to see if you connected to the page directly by typing the URL into the browser. If you connect to the page this way, a "referer" is not found in the header and the browser is redirected to the main page that contains the frameset.

The second half of the If statement is (Left(Request.ServerVariables("HTTP_REFERER"),42) <> "http://www.myserver.com/AppDir/mainfrm.htm"). This line checks to see that you connected to frame1.asp from the frameset page(mainfrm.htm). If not, the browser is redirected to the main page containing the frameset.

The line:

```
Response.Redirect "http://www.myserver.com/AppDir/mainfrm.htm"\
```

redirects the browser to the following page:

```
http://www.myserver.com/AppDir/mainfrm.htm.
```

You can use the following code to demonstrate this. Save each in a separate file and use the names indicated below. Also, you need to make the following modifications to the code in the frame1.asp file:

1. Change the two instances of http://www.myserver.com/AppDir/mainfrm.htm in the If statement to point to the correct virtual root and directory on your server.
2. Change the value in the Left() function from 42, to the number of characters in the path you just modified in step 1.

File: Mainfrm.htm

```
<HTML>
   <HEAD><TITLE>MAINFRM</TITLE></HEAD>
   <BODY>
   <FRAMESET ROWS="400,*">
      <FRAME SCROLLING="no" NORESIZE SRC="frame1.asp">
      <FRAME SCROLLING="no" NORESIZE SRC="frame2.asp">
   </FRAMESET>
   </BODY>
   </HTML>
```

File: Frame1.asp

```
<%
   If (Request.ServerVariables("HTTP_REFERER") = "") Or
   (Left(Request.ServerVariables("HTTP_REFERER"),42) <>
   "http://www.myserver.com/AppDir/mainfrm.htm") Then
      Response.Redirect "http://www.myserver.com/AppDir/mainfrm.htm"
   End If
   %>

   <HTML>
   <HEAD><TITLE>FRAME1</TITLE></HEAD>
   <BODY>
   In Frame 1.
   </BODY>
   </HTML>
```

File: Frame2.asp

```
<HTML>
   <HEAD><TITLE>FRAME2</TITLE></HEAD>
   <BODY>
   In Frame 2.
   </BODY>
   </HTML>
```

KBCategory: kbprg kbhowto
KBSubcategory: AXSFHTML
Additional reference words: 1.00 kbdsi

Article ID: Q157748
Creation Date: 17-OCT-1996
Revision Date: 27-JAN-1997
The information in this article applies to:

® Microsoft Active Server Pages, version 1.0

Summary

Internet Information Server (IIS) Active Server Pages (ASP) can be used to modify information in an ActiveX Layout file (.alx). To do this, server side scripting should be placed in the .alx file.

More Information

Active Server Pages are run when the URL is processed by IIS. At this time, the controls in an .alx file have not been created in Internet Explorer (IE). Because of this, it is impossible to change a control in an .alx file from the HTTP document hosting the .alx file using server-side scripting.

To do this, place the server-side scripting code in the .alx file itself. When IE calls for the .alx file, IIS processes the server-side script before handing the .alx file back to IE. The key to making this all happen is to rename the .alx file with an ASP extension.

Note The following example assumes that the Adventure Works sample site is installed and working. It draws data from the AdvWorks data source.

To see an example of how this all works, use the following steps:

1. Using the ActiveX Control Pad, create a new HTML Layout (.alx).
2. Add a label (Label1) and a text box (TextBox1) control to the layout file.
3. Click on the text box, and type "test" in the text box without the quotes.
4. On the File click Save As.
5. Save the file as "layout1.asp" without the quotes. Note that this is NOT the default extension of .alx. Make sure you save all files in this example in the same directory. This directory should be a directory published by IIS, such as wwwroot. Make sure that IIS has execute privileges on this directory so that it can run the script. The root of the wwwroot directory does not have execute permissions setup by default.
6. Using the ActiveX Control Pad, create a new HTML document.
7. On the Edit menu, click Insert HTML Layout.
8. Type in the name "layout1.asp" in the file name text box.

   Note This file does not show up in the list of files.

   ---

9. An object tag similar to the following appears in your HTML document:

   ```
   <OBJECT CLASSID="CLSID:812AE312-8B8E-11CF-93C8-00AA00C08FDF"
   ID="layout1_alx" STYLE="LEFT:0;TOP:0">
   <PARAM NAME="ALXPATH" REF VALUE="layout1.asp">
   </OBJECT>
   ```

   If the VALUE= contains a full path to the layout1.asp file, remove it so itreads as above.
10. Save this file as "page1.htm".
11. Open the file layout1.asp in a text editor such as Notepad. You cannot open this in the ActiveX Control Pad because it brings up the layout editor.
12. Add the following lines at the beginning of the file:

    ```
    <%
    Set Conn = Server.CreateObject("ADO.Connection")
    ```

```
Conn.Open "AdvWorks"
rs = Conn.Execute("SELECT * FROM products")
%>
```

13. Replace the line for the label object that reads as follows:

    ```
    <PARAM NAME="Caption" VALUE="label1">
    ```

    With the following line:

    ```
    <PARAM NAME="Caption" VALUE="<% Response.Write rs(1).Name %>">
    ```

15. Replace the line for the text box object that reads as follows:

    ```
    <PARAM NAME="Value" VALUE="test">
    ```

    With the following line:

    ```
    <PARAM NAME="Value" VALUE="<% Response.Write rs(1) %>">
    ```

16. Save "layout.asp."
17. Open Internet Explorer and view page1.htm.

Warning After adding server-side script to your .alx file, be careful about editing the file in the ActiveX Control Pad. If you open the file in the ActiveX Control Pad, it converts all <% and %> markers to <% and %&gt, respectively. This breaks the server-side script. You need to reverse all of these changes manually. Therefore, it is best to have your .alx file completely designed before adding server-side script to the file.

KBCategory: kbprg kbhowto
KBSubcategory: AXSFHTML AXSFVBS
Additional reference words: 1.00 kbdsi

Article ID: Q163010
Creation Date: 31-JAN-1997
Revision Date: 31-JAN-1997

The information in this article applies to:

® Microsoft Active Server Pages, version 1.0

Summary

Active Server Pages (ASP) uses cookies to track SessionIDs. As a result, the browser is sent a cookie when a new session is created. Typically this occurs the first time a particular client requests an ASP page in your Web application. If Internet Explorer is configured to "Warn before accepting cookies," you will see a warning dialog box when this SessionID cookie is sent to the browser. This article explains how to prevent ASP from sending these cookies.

Note Disabling SessionID cookies is not recommended because it seriously limits the functionality of Active Server Pages.

More Information

Active Server Pages provides a configurable registry setting that can be used to disable Session State. The following registry value controls Session State:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\ASP\
    Parameters\AllowSessionState
```

This setting is not present in the registry by default. To disable the sending of cookies, you need to add this as a new binary value in the registry editor (Regedit.exe).

If you set AllowSessionState to 0, ASP no longer sends SessionID cookies to the browser.

Warning Setting AllowSessionState to 0 disables the use of sessions in your Web application. This means that you will not be able to store or retrieve session variables. If your Web application uses session variables, as most do, your pages will no longer function properly. In addition, you will encounter scripting errors in the server-side scripts, which use session variables.

KBCategory: kbprg kbhowto
KBSubcategory: AXSFHTML
Additional reference words: 1.00

Article ID: Q159325
Creation Date: 12-NOV-1996
Revision Date: 02-JAN-1997

The information in this article applies to:

® Microsoft Access 97

Summary

Moderate: Requires basic macro, coding, and interoperability skills.

Microsoft Access 97 provides features that enable you to publish data from your database as Web pages. You can create both static and dynamic Web pages; however, each type of Web page may have different requirements for the server that stores the page and the browser that views it. This article summarizes those requirements.

More Information

When you click Save As HTML on the File menu in Microsoft Access 97, you start the "Publish to the Web" Wizard, which helps you publish Web pages in three different formats: Static HTML, Dynamic HTX/IDC (Internet Database Connector), and Dynamic ASP (Microsoft Active Server Pages). Each format uses a different technology in the Web pages you create, and each technology may have specific requirements for your Web server and browser software.

The following table summarizes the requirements for each Web page format:

File Types	Specific Server?	Specific Details?	Details
Static HTML (*.htm, *.html)	No	No	Static Data; supported by all Web server and browser software
HTX/IDC (*.idc, *.htx)	No	Yes	Dynamic data; requires one of the following on the server: Microsoft Internet Information Server (IIS), version 1.x with Internet Database Connector add-in; IIS version 2.0 or later, Active Server on Windows NT Server, version 4.0; Microsoft Personal Web Server on Windows 95; or Windows NT Workstation, version 4.0.
ASP (*.asp)	Yes	No	Dynamic data; requires one of the following on the server: Microsoft IIS, version 3.0 or later on Windows NT Server, version 4.0; Microsoft Personal Web Server on Windows 95 with Active Server; or Microsoft Peer Web Services with Active Server on Wiindows NT Workstation, versin 4.0.

References

For more information about creating Web pages for your database, search the Help Index for "Publish to the Web Wizard," or ask the Microsoft Access 97 Office Assistant.

If you create ASP files based on a form in a Microsoft Access database, you must browse the form using Microsoft Internet Explorer, version 3.0 or later with the HTML Layout ActiveX control; you can browse other Microsoft Access database objects in ASP file format using any Web browser.

Article ID: Q159682
Creation Date: 19-NOV-1996
Revision Date: 29-NOV-1996

The information in this article applies to:

® Microsoft Access 97

Symptoms

Moderate: Requires basic macro, coding, and interoperability skills.

After you use the "Publish to the Web" Wizard to create Dynamic HTX/IDC or Dynamic ASP (Microsoft Active Server Pages) file formats on your Web server, you may receive the following error message when you try to view those files with your Web browser:

> Data source name not found and no default driver specified

The relative path and file name of the page you tried to open are displayed beneath the error message.

Cause

You did not create a System DSN data source on the server, or your Web page file contains an incorrect reference to the System DSN data source name.

Resolution

Follow these steps on your Web server computer to add a new System DSN or check the name of an existing data source:

1. Double-click the ODBC icon in Control Panel on your Web Server.
2. In the Data Sources dialog box, click System DSN.
3. Click Add if you do not see the name of the System DSN you used as the Data Source Name in the "Publish to the Web" Wizard.
4. Select Microsoft Access Driver, and then click Finish.

Note If the Microsoft Access Driver does not appear, it is not installed on your Web server. For information about installing the driver on your Web server, search the Help Index for "Microsoft Access Desktop driver," or ask the Microsoft Access 97 Office Assistant.

---

5. Complete the ODBC Microsoft Access 97 Setup dialog box. The name you type in the Data Source Name box is the name you will use in the Data Source Name box in the "Publish to the Web" Wizard.
6. Click OK to close the ODBC Microsoft Access 97 Setup dialog box.
7. Click OK in the ODBC Data Source Administrator dialog box.
8. If the Data Source Name in the System DSN on your Web server is different from the one you used when you created your web pages, start the "Publish to the Web" Wizard in Microsoft Access and recreate your web pages using the correct Data Source Name.

More Information

Dynamic HTX/IDC and Dynamic ASP file formats are Web pages that use ODBC to display updated information from your database each time you open or refresh the page in your Web browser.

Note Dynamic HTX/IDC and Dynamic ASP files require specific Web server software, and in some instances specific Web browser software. For information about the requirements to use ASP files, please see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q159325**
TITLE: Server and Browser Requirements for Publish To Web Wizard

Steps to Reproduce Behavior

1. Open the sample database Northwind.mdb.
2. On the File menu, click Save As HTML to start the "Publish to the Web" Wizard.
3. In the wizard's first dialog box, click Next.
4. In the "What do you want to publish?" dialog box, click the Customers table, and then click Next.
5. In the "What HTML document do you want to use as a default template?" dialog box, click Next.
6. In the "What default format type do you want to create?" dialog box, click Dynamic ASP, and then click Next.
7. In the "What are, or will be, the setting for the Internet database?" dialog box, type MyDSN in the Data Source Name box, and type the server URL of the location where you will store the ASP file in the Server URL box. Click Next.
8. In the "I want to put my Web publication in this folder" box, type the full path to the folder on your Web server where you will store the ASP file. Click Finish.

   Note If you do not have write permissions for the destination folder on your Web server, you receive the following error message:

   Microsoft Access can't save the output data to the file you've selected.

9. Start your Web browser, and type the URL for the Customers_1.asp file you just created, for example, http://MyServer/MyFolder/Customers_1.asp. Note that you receive the error message:

   Data source name not found and no default driver specified

References

For more information about Active Server Pages and HTX/IDC file formats, search the Help Index for "dynamic HTML format," or ask the Microsoft Access 97 Office Assistant.

Article ID: Q161779
Creation Date: 03-JAN-1997
Revision Date: 09-JAN-1997

The information in this article applies to:

® Microsoft FrontPage 97 for Windows

Summary

Microsoft Internet Information Server (IIS) Active Server Pages (ASP) pose unique challenges to authors using FrontPage. This article discusses some important considerations about using FrontPage 97 to manage or edit Active Server Pages.

Note Earlier versions of the FrontPage Server Extensions do not support IIS ASP syntax. To use Active Server Pages, install the FrontPage 97 Server Extensions for the Microsoft Internet Information Server.

More Information

FrontPage 97 allows you to create and manage active server pages. In general, when you create Active Server Pages in FrontPage Editor, be aware of the following considerations:

1. FrontPage can not manage links which originate in a Meta tag. It is customary to create links to Active Server Pages by using the http-equiv"REFRESH" attribute of the Meta tag. However, links that originate in a Meta tag do not appear as links in the FrontPage Explorer and they are not updated when you rename or move the files that link to or from these links.
2. Save the file using the .asp file name extension. IIS 3.0 requires the .asp file name extension in order to process Active Server Pages.
3. Save the file in an executable directory. Files that are stored in a non-executable directory do not execute. For example, when you view the page in a web browser, the page is not interpreted and rendered correctly by the web browser and the code is displayed instead of its result.

   Note You cannot browse an executable directory. If you mark a directory that contains HTML pages as executable, users will not see your content.

   ---

4. FrontPage Editor treats ASP syntax as a text-level object and requires that it be embedded within one of the following HTML elements:

   ```
   paragraph (<p>)
   table row and table cell (<tr>, <td>)
   list (<li>)
   ```

   On the other hand, if the ASP is the value of a tag attribute, then the ASP does not need to be embedded within an HTML element. (See item 6 below.)

   If the ASP code is not embedded inside an HTML tag or as a tag attribute, the FrontPage Editor automatically inserts the opening and closing tags around the ASP code. In this case, FrontPage also automatically removes these tags when you save the page or when you click the HTML command on the View menu. Although this process does not adversely affect most HTML, FrontPage may rearrange the syntax of your ASP code in such a way that it ceases to function properly. In this case, you may need to embed the affected code inside an HTML element, such as a paragraph, table, or list element using the View Or Edit HTML dialog box. (To access this dialog box, click HTML on the View menu.)
5. .Active server pages can contain scripted links, as in this example:

   ```
   <a href="/<%OBJnextlink.GetNextUrl("/www/nextlink.txt")%></a>
   ```

   FrontPage Explorer reports these links as broken; however, the links function if you post your content on a properly configured IIS 3.0 server.
6. FrontPage Editor maintains an internal list of HTML tags which allow extended attributes. If you attempt to

insert ASP code inside a tag that does not allow extended attributes, the ASP code is discarded by the FrontPage Editor. For additional information, please see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q161420**
TITLE: FrontPage Editor Deletes Unknown Attributes in HTML

7. Server-side includes do not expand in the FrontPage Editor. They are parsed into an HTML markup section. They still function if posted on a properly configured IIS 3.0 server. The following is an example of a server side include:

```
<!--#include virtual="/test.inc"-->
```

Note An HTML markup section is created from the View Or Edit HTML dialog box.

---

8. Multiple line link structures may be stripped out. Active Server Pages can process multi-line scripts to evaluate the correct URL to render in a web browser. FrontPage does not parse an HREF attribute on an Anchor tag if the HREF contains either a carriage return or a line feed character. To preserve this code, type the code in the View Or Edit HTML dialog box.

Additional Information

To create Active Server Pages in FrontPage Editor, use the following steps:

1. On the Insert menu, click Script.
2. Under Language, select VBScript, and then click to select Run Script On Server
3. In the Script text box, type the ASP code.
4. Click OK.

FrontPage will automatically surround your code with the "<%" and "%>" characters to enable execution on an IIS 3.0 server.

Note You can also type the server code in the View Or Edit HTML dialog box, but you must manually add the < % and % > characters to enable execution on an IIS 3.0 server.

FrontPage inserts the Visual Basic Script icon in your document to indicate that a Visual Basic script has been added to your page. You can view or edit the script by double-clicking the icon.

You may also add scripts to the attributes of HTML tags if the tags support extended attributes. Any tag which supports extended attributes will include an Extended button in its Properties dialog box. In this case, a Visual Basic Script icon is not displayed in the FrontPage Editor.

Article ID: Q162145
Creation Date: 14-JAN-1997
Revision Date: 15-JAN-1997

The information in this article applies to:

® Microsoft FrontPage 97 for Windows

Summary

This article describes how to configure FrontPage 97 Server Extensions for Windows NT-based web servers by modifying parameter setting in the Frontpg.ini file.

More Information

The Frontpg.ini is located in the Windows NT directory, which is typically C:\Winnt. If the server is multi-homed, the sections for each virtual server are:

® [Port <IPAddress>:<port>] on an IIS server, and

® Port <Hostname>:<port>] on other NT servers

Global settings — those that affect all virtual servers — are stored in the [FrontPage 2.0] section of the Frontpg.ini file. The default setting for each parameter is 0, unless otherwise noted.

The parameter syntax is as follows:

```
Parameter=value
```

When you place the parameter in the [FrontPage 2.0] section of the Frontpg.ini file, all ports and all web servers installed on that server are affected by the setting. When you place the parameter in the [Port *:<port>] section of the Frontpg.ini file, the specific port is affected.

Logging

When you set Logging to a non-zero value, authoring operations are logged in the Author.log file located in the _vti_log folder of the root web. Each log entry includes the current time, the remote host, the author's user name, the name of the web, the operation performed, and per-operation data.

NoSaveResultsToAbsoluteFile

When you set NoSaveResultsToAbsoluteFile to a non-zero value, you prevent the Save Results, Registration, and Discussion WebBot components from writing to an absolute file path when the browsing account has the NTFS rights set to write to that path. When this is set, the Save Results WebBot component can only write a file within the web's content area.

NoExecutableCgiUpload

When you set NoExecutableCgiUpload to a non-zero value, authors cannot upload files to a directory marked as executable and they cannot mark a directory as executable. For example, when you set NoExecutableCgiUpload to a non-zero value, authors are prevented from uploading and executing Active Server Pages (ASP), Internet Database Connector Pages (IDC), PERL Scripts (PL), CGI scripts, and ISAPI extensions. To restrict an author's ability to upload and execute CGI scripts and ISAPI extensions only, use the AllowExecutableScripts configuration parameter.

AllowExecutableScripts

When you set AllowExecutableScripts to a non-zero value, FrontPage sets the executable bit on files within executable directories. Hence, when directories are marked as executable, all files within the directory are also marked as executable. For example, when you set this parameter to a non-zero value, authors will be able to execute newly uploaded CGI scripts and ISAPI extensions because they will be marked executable.

ReformatHtml

When you set ReformatHtml to "Y" or a non-zero value, the FrontPage 97 Server Extensions reformat all HTML pages as they are uploaded. When you set ReformatHtml to zero, the FrontPage 97 Server Extensions reformat only pages that contain WebBot components.

UpperCaseTags

When you set UpperCaseTags to "Y" or a non-zero value, all HTML tags are converted to upper-case characters when the FrontPage 97 Server Extensions reformat the HTML pages.

PreserveTagCase

When you set PreserveTagCase to "Y" or a non-zero value, the case of HTML tag attributes is preserved when the FrontPage 97 Server Extensions reformat the HTML pages. Note that the tag itself will always be upper- or lower-case according to the UpperCaseTags attribute.

TextMemory

When you set TextMemory to zero, you turn off full-text indexing of your webs. When you set TextMemory to a non-zero value, you allocate how many megabytes the full-text indexing uses for its hash-tables and other data structures. The default setting is 1.

If the Web Project Wizard can’t connect to the specified Web server, that server’s FrontPage Server extensions may not be set up correctly. Visit the Visual InterDev Web site at http://www.microsoft.com/vinterdev and refer to the Technical FAQ to help you troubleshoot common setup issues.

Article ID: Q163034
Creation Date: 31-JAN-1997
Revision Date: 06-FEB-1997

The information in this article applies to:

® Microsoft Access 97

Symptoms

Advanced: Requires expert coding, interoperability, and multiuser skills.

When you browse to an ASP file that was exported with Microsoft Access 97, the Format property of fields in Microsoft Access forms is not preserved. This article discusses the workaround for this behavior.

Cause

When forms are exported to ASP format, Microsoft Access 97 generates a SQL statement for the ASP file that retrieves the data from the underlying tables. This functionality is not designed to preserve Format properties of fields or form controls. Numbers will be displayed in General Number format, Dates will be displayed in General Date format, Times will be displayed in Long Time format, and Yes/No fields will be displayed in True/False format.

Resolution

In order to preserve the Format properties of fields when exporting tables and queries to ASP format, one workaround is to export a Microsoft Access query that uses the Format function around the fields that you want formatted. However, this workaround should not be used for forms. When exporting forms to ASP format, the fields that use the Format function will not be updateable and ASP forms are designed for use with updateable recordsets.

Note This section contains information about editing ASP files and assumes that you are familiar with editing HTML files, Active Server, and Visual Basic Scripting. Microsoft Access Technical Support engineers do not support customization of any HTML, HTX, IDC, or ASP files.

---

1. Open the sample database Northwind.mdb in Microsoft Access 97.
2. Use the AutoForm: Columnar Wizard to create a new form based on the Products table.
3. Switch to Design view of the form, delete the ProductID label and text box because it is an AutoNumber field that you do not want users to enter data in. Save the form and call it frmFormatTest. Close the form.
4. Select frmFormatTest in the Database window.
5. On the File menu, click Save As/Export.
6. In the Save As dialog box, click to select "To an External File or Database," and click OK. Note that the "Save Form 'frmFormatTest' In" dialog box appears.
7. In the Save As Type box, select Microsoft Active Server Pages (*.asp) and type frmFormatTest.asp in the File Name box. Note the folder where the files will be exported to. Click Export. Note that the Microsoft Active Server Pages Output Options dialog box appears.
8. In the Data Source Name box, enter the name of a System DSN that points to the sample database Northwind.mdb.

   For more information about how to define a system DSN, search the Help index for "ODBC, setting up data sources," and see the following article in the Microsoft Knowledge Base:

   ARTICLE-ID: **Q159682**
   TITLE: "Data Source Name Not Found" Err Msg Opening Web Page
9. In the Server URL box, enter the URL that points to the Web Server location where your ASP files will be stored. For example, if you store the ASP files in the \ASPsamp folder on the \\PubTest server, type "http://pubtest/aspsamp/" (without the quotation marks) as your Server URL. Click OK. The ASP output creates the files frmFormatTest.asp and frmFormatTestalx.asp.

10. Use Notepad or another text editor to open the frmFormatTestalx.asp file. Towards the middle of the file, you will find the following code which sets the value of the UnitPrice object equal to the value of the current UnitPrice field in the recordset.

```
      <OBJECT ID="UnitPrice"
      CLASSID="CLSID:8BD21D10-EC42-11CE-9E0D-00AA006002F3"

STYLE="TOP:225;LEFT:168;WIDTH:231;HEIGHT:25;TABINDEX:5;ZINDEX:8;">
      <%If Not IsNull(rs.Fields("UnitPrice").Value) Then%>
      <PARAM NAME="Value"
      VALUE="<%=Server.HTMLEncode(rs.Fields("UnitPrice").Value)%>">
      <%End If%>
      <PARAM NAME="BackStyle" VALUE="1">
      <PARAM NAME="BackColor" VALUE="2147483653">
      <PARAM NAME="BorderStyle" VALUE="1">
      <PARAM NAME="BorderColor" VALUE="0">
      <PARAM NAME="ForeColor" VALUE="2147483656">
      <PARAM NAME="FontHeight" VALUE="160">
      <PARAM NAME="Font" VALUE="MS Sans Serif">
      <PARAM NAME="FontName" VALUE="MS Sans Serif">
      <PARAM NAME="Size" VALUE="6006;650">
      <PARAM NAME="SpecialEffect" VALUE="2">
      <PARAM NAME="VariousPropertyBits" VALUE="2894088219">
      </OBJECT>
```

To format the value as Currency, you should use the VB Script formatcurrency() function by modifying the <PARAM NAME="Value"...> so that it matches the following:

```
      <PARAM NAME="Value"

VALUE="<%=Server.HTMLEncode(formatcurrency(rs.Fields("UnitPrice")
      .Value))%>">
```

Note VB Script has the FormatCurrency, FormatNumber, FormatDatetime, and FormatPercent functions, but there is no Format function with which you can provide custom formats. Please refer to your VB Script Language Reference available in the ASP online documentation for more information on VB Script functions.

Save and then close frmFormatTestalx.asp.

11. Copy frmFormatTest.asp and frmFormatTestalx.asp to a folder on your Web Server computer where you have Execute permission. For more information about configuring Microsoft Internet Information Server (IIS) permissions, please refer to the IIS Help Index, and see the following article in the Microsoft Knowledge Base:

    ARTICLE-ID: **Q162975**
    TITLE: Permissions Necessary to View HTML, IDC, and ASP Files

12. Start Microsoft Internet Explorer 3.0, or another Web browser program.
13. Type the Uniform Resource Locator (URL) in the address box of your Web browser to view frmFormatTest.ASP. For example, if you saved your ASP file in a folder called Test in the wwwroot folder of your Web Server, type:

    http://<servername>/test/frmFormatTest.ASP

    Note that the URL depends upon where your files are located on the Web Server.

14. Note that the UnitPrice field has the Currency format applied.

References

For more information about exporting forms to ASP, search the Help Index for "ASP files," and then "Export a form to dynamic HTML format."

```
'if user isn't requesting the profile page, redirect them there
startPage = "/StateU/profile.asp"
currentPage = Request.ServerVariables("SCRIPT_NAME")

If strcomp(currentPage,startPage,1) Then
    Response.Redirect(startPage)
End If
```

```
currentPage = Request.ServerVariables("SCRIPT_NAME")
Session("requestedPage") = currentPage
```

```
<%
If Not (IsEmpty(Request("txtID"))) Then
    'request came from Submit button on form
    'save form data into Session variables
    Session("username") = Request("txtName")
    Session("id") = Request("txtID")
    Session("major") = Request("Major")

    'redirect to originally requested page
    If InStr(Session("requestedPage"), "profile") = 0 Then
        Response.Redirect Session("requestedPage")
    Else
        Response.Redirect "default.htm"
    End If
End If
%>
<HTML>
...
```

Article ID: Q163443
Creation Date: 10-FEB-1997
Revision Date: 11-FEB-1997

The information in this article applies to:

® Microsoft Access 97

Symptoms

When you try to save an action query in HTML, IDC, or ASP format, you may receive one of the following error messages:

® Query input must contain at least one table or query.

® You can only save select, crosstab, or union queries to this format.

® An action query cannot be used as a rowsource.

Cause

Action queries cannot be exported to HTML, IDC, or ASP format.

Resolution

If you want to generate Internet Database Connector (IDC) files that perform action queries, you must write the IDC file manually using Notepad or another text editor. Following is an example of an IDC file that inserts a record into a Microsoft Access 97 database.

Instead of exporting an action query to Active Server Pages (ASP) format to update your database, consider exporting a form to ASP format. Forms in ASP format support adding, updating, and deleting records.

Status

This behavior is by design.

More Information

Following is an example of how to create an HTML Insert Form that uses IDC files to insert records into a Microsoft Access 97 database.

Note This section contains information about editing IDC files, and assumes that you are familiar with editing IDC files. Microsoft Access Technical Support engineers do not support customization of any HTML, HTX, IDC, or ASP files.

Creating the Files in Microsoft Access

1. Start Microsoft Access 97 and open the sample database Northwind.mdb.
2. Create the following new query called EnterShipper based on the Shippers table:

Query	**EnterShipper**
Type:	Select Query
Field:	CompanyName
Table:	Shippers

3. On the Query menu, click Parameters.
4. Type the following in the Query Parameters dialog box, and then click OK.

Parameter	**Data Type**
[EnterName]	Text

5. Save the EnterShipper query and close it.
6. Select the EnterShipper query in the Database window, and then click Save As/Export on the File menu.
7. In the Save As dialog box, click "To an External File or Database," and then click OK.
8. In the "Save Query 'EnterShipper' In" dialog box, select Microsoft IIS 1-2 (*.htx;*.idc) in the Save As Type box, and type EnterShipper.htx in the File Name box. Note the folder where the exported files will be stored. Click Export.
9. In the HTX/IDC Output Options dialog box, type the name of a System DSN on your Web server that points to the Northwind sample database. Click OK.

   For more information on how to define a system DSN, search the Help index for "ODBC, setting up data sources," and see the following article in the Microsoft Knowledge Base:

   ARTICLE-ID: **Q159682**
   TITLE: Data Source Name Not Found" Err Msg Opening Web Page
10. Click OK in the Enter Parameter Value dialog box that appears.
11. The HTX/IDC output creates three files: EnterShipper.HTML, EnterShipper.htx, and EnterShipper.IDC.
12. Copy EnterShipper.HTML, EnterShipper.htx, and EnterShipper.IDC to a folder on your Web server computer where you have both Read and Execute permission. Read permission is necessary to browse the HTML file, and Execute permission is necessary to run the IDC file.

    For more information about configuring Microsoft Internet Information Server (IIS) permissions, please refer to the IIS Help Index, and see the following article in the Microsoft Knowledge Base:

    ARTICLE-ID: **Q160754**
    TITLE: Error "HTTP/1.0 403 Access Forbidden" Browsing IDC Page

Customizing the IDC files to Permit Inserting Records

1. Use Notepad or another text editor to open the EnterShipper.IDC file. You need to change the SQL Statement so that it will use the value passed from the HTML form and insert it into the Shippers table of the Northwind sample database. Change the SQL Statement so that it looks like this:

   ```
   INSERT into Shippers (CompanyName) values ('%[EnterName]%');
   ```

2. Save the EnterShipper.IDC file and close it.
3. Start Microsoft Internet Explorer 3.0, or another Web browser program.
4. Type the Uniform Resource Locator (URL) in the address box of your Web browser to view EnterShipper.HTML. For example, if you saved your IDC files in a folder called Test in the wwwroot folder of your Web server, type:

   http://<servername>/test/EnterShipper.HTML

   Note that the URL depends upon where your files are located on the Web Server.
5. The EnterShipper.HTML form opens in your Web browser with an [EnterName] box and a Run Query button. Type a name into the box, and then click the Run Query button. A new record with the name you typed is inserted into the Shippers table in the Northwind sample database. Note that no records are returned to your web browser because an insert query does not return records.

References

For more information about exporting IDC files, search the Help Index for "IDC files," or ask the Microsoft Access 97 Office Assistant. In addition, please refer to the Microsoft Internet Information Server (IIS) Help Index.

For more information about exporting ASP files, search the Help Index for "ASP files," or ask the Microsoft Access 97 Office Assistant. In addition, please refer to your online ASP documentation which is installed when you install IIS 3.0.

Article ID: Q163510
Creation Date: 11-FEB-1997
Revision Date: 12-FEB-1997

The information in this article applies to:

® Microsoft Access 97

Symptoms

Novice: Requires knowledge of the user interface on single-user computers.

The Microsoft Access 97 Help topic for the OutputTo method incorrectly states that the intrinsic constant to output Active Server Pages (ASP) files is acFormatActiveXServer.

Resolution

If you want to output ASP files, use the intrinsic constant acFormatASP as the Outputformat argument of the OutputTo method.

Article ID: Q163603
Creation Date: 12-FEB-1997
Revision Date: 19-FEB-1997

The information in this article applies to:

® Microsoft Access 97

Summary

TestDSN.htm and TestDSN.asp are Web files that you can copy to your Web server. Use them to test a System DSN on your Web server that connects to a Microsoft Access 97 database. If the System DSN is valid, you receive a record count from a table in your Microsoft Access 97 database. If the System DSN is not valid, you receive information that you can use to troubleshoot the connection.

~ TestDSN.exe is available for download from the Microsoft Software Library. It contains the following files:

TestDSN.htm	883 bytes	02-12-97	Static HTML File
TestIDSN.asp	564 bytes	02-12-97	Active Server Pages File
Readme.txt			Usage and installation instructions

For more information about downloading files from the Microsoft Software Library, please see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q119591**
TITLE: How to Obtain Microsoft Support Files from Online Services

```
Sub Application_OnStart
    Application("hitcounter") = 0
End Sub

Sub Session_OnStart
    Application.Lock
    Application("hitcounter") = Application("hitcounter") + 1
    Application.Unlock
    ...
End Sub
```

```
Sub Application_OnEnd
    Dim fs, txtFile
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set txtFile = fs.CreateTextFile("c:\hitcounter.txt", True)
    txtFile.WriteLine(Application("hitcounter"))
    txtFile.Close
End Sub
```

```
Sub Application_OnStart
    Dim fs, txtFile
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set txtFile = fs.OpenTextFile("c:\hitcounter.txt")
    Application.Lock
    Application("hitcounter") = txtFile.ReadLine
    Application.Unlock
    txtFile.Close
End Sub
```

Article ID: Q162976
Creation Date: 31-JAN-1997
Revision Date: 31-JAN-1997

The information in this article applies to:

® Microsoft Access 97

Summary

This article describes the steps to test your Web server to make sure that the Active Server Pages(ASP) feature is working correctly.

More Information

The following steps walk you through opening the Active Server Pages Roadmap and executing one of the ADO sample pages.

Note You must be at the Server computer for these steps to work

---

u **Steps to Test Your Server**

1. Click the Start button, point to Programs, point to Microsoft Internet Server or Microsoft Personal Web Server, and click Active Server Pages Roadmap.
2. Select the More Samples hyperlink from the ASP In Action column.
3. Click the "ADO using Server.CreatObject" hyperlink.

Results

You should see table returned with Order information. If you see the table, then both Active Server Pages and ADO are working correctly.

If you did not see the table, you need to verify that Active Server Pages was installed correctly.

References

For additional help with installing Active Server Pages, please see the Active Server Pages Web site on the Microsoft World Wide Web at the following address,

`http://www.microsoft.com/iis/LearnAboutIIS/ActiveServer/default.asp`

visit the following newsgroup,

msnews.microsoft.public.inetserver.iis.activeserverpages

or contact Internet Information Server Support.

For additional information on Personal Web Server, please see the Personal Web Server Site on the Microsoft World Wide Web site at the following address:

`http://www.microsoft.com/ie/iesk/pws.htm`

Article ID: Q164001
Creation Date: 20-FEB-1997
Revision Date: 21-FEB-1997

The information in this article applies to:

® Microsoft Access 97

Summary

Moderate: Requires basic macro, coding, and interoperability skills.

If your Web server is running Microsoft Peer Web Server for Windows NT 4.0 or Microsoft Personal Web Server for Windows 95, you must also load Active Server Pages (ASP) if you want to view ASP files created in Microsoft Access 97.

Note If your Web server is running Microsoft Internet Information Server (IIS), your must have version 3.0 in order to view ASP files. You can download IIS version 3.0 from http://www.microsoft.com/iis/.

This article details the steps to download Active Server Pages from the Microsoft Internet Information Server (IIS) home page on the Internet.

More Information

Active Server Pages is an additional feature of IIS 3.0 that also works with Personal Web Server on Microsoft Windows 95 and Peer Web Server on Microsoft Windows NT Workstation version 4.0.

Follow these steps to download the Active Server Pages component from Microsoft's Web site:

1. Start your Web browser and type the following Uniform Resource Locator (URL) in the address line:

   ```
   http://www.microsoft.com/iis/
   ```

2. Click the Get IIS button, and then click Download IIS 3.0.
3. From the Internet Information Server 3.0 Download page, click the link to Register to Download.
4. Fill out the Registration form, and then click the Submit User Info button.
5. On the "Choose the IIS 3.0 Features You Would Like to Download" page, clear the check boxes for all components except Active Server Pages, and select the correct language version. Click Next.
6. On the "Choose A Download Location" page, pick a location from which to download the file, and then click "Download from this site."
7. On the "Click On Each Item Below In Turn To Download" page, click the Active Server Pages link to open or download Asp.exe, and install Active Server Pages.

References

For more information about creating ASP files in Microsoft Access, search the Help Index for "ASP files," or ask the Microsoft Access 97 Office Assistant.

To complete the lab exercises in this chapter, you must have the required software. For detailed information about the labs and setup for the labs, click Labs in this course.

For background information on this lab, click each of these topics:

Objectives

Prerequisites

Lab Setup

To see a demonstration of the lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

To see a diagram of how the files you edit in this lab will fit into the State University Web site, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

Estimated time to complete this lab: **90 minutes**

Note There are project and solution files associated with this lab. If you installed the labs during Setup, these files are in the folder <*Install Folder*>\Labs\Lab09 on your hard disk. If you did not install the labs during Setup, you can find them in the \Labs\Lab09 folder of this *Mastering Web Site Development* CD-ROM.

Exercises

The following exercises provide practice working with the concepts and techniques covered in Chapter 9: Using Microsoft Transaction Server.

Exercise 1: Creating the State University Package

In this exercise, you will create a new package in Microsoft Transaction Server that contains all of the components for State University. You will then package the **Add** component, and test it by calling it from an .asp file.

Exercise 2: Adding Transaction Support

In this exercise, you will modify the **Drop** object to call **SetComplete** or **SetAbort** to support transactions in Microsoft Transaction Server. You will also modify the **Transfer** object to create the **Add** and **Drop** objects using **CreateInstance**.

The **Transfer** object will then call the **Add** and **Drop** objects to perform the work of transferring a student. Finally, you will add the **Drop** and **Transfer** objects to the new package, and then test the package by calling the objects from .asp files.

After completing this lab, you will be able to:

- Create a new package by using Microsoft Transaction Server Explorer.
- Add components to a package, and set their properties by using Microsoft Transaction Server Explorer.
- Monitor transaction statistics by using Microsoft Transaction Server Explorer.
- Create a component that supports MTS transactions by calling either the **SetComplete** method or the **SetAbort** method.
- Use the MTS **Context** object to create other components in the same transaction.

Before completing this lab you must be able to:

® Create Web pages, .asp files, and projects in Microsoft Visual InterDev.

® Call an ActiveX server component from an .asp file.

® Create an ActiveX server component by using Visual Basic 5.0

To complete this lab, you need the following:

® The Visual InterDev State University Project
® The State University SQL database
® Microsoft Transaction Server
® Visual Basic 5.0

In this exercise, you will create a new package in Microsoft Transaction Server that will contain all of the ActiveX server components for the State University Web site. You will add the **Add** component to the new package, and then test the package by calling it from an .asp file.

You will use the classview.htm file to add and drop classes. To see an illustration of how classview.htm looks, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

When you add a class using the classview.htm file, the addclass.asp file calls the **Add** object to add a student to a class. To see an illustration of how addclass.asp looks when returned to the student, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

u **Create the State University package**

1. Open the Microsoft Transaction Server Explorer.
2. Create a new package named State University. When asked to set the package identity, set it to **Interactive user**.

   For more information on creating packages, see Creating a Package.

u **Add a component to the State University package**

1. Create a folder on the Web server named \busobjects.
2. Copy the following files from the \MWD\Labs\Lab09 folder to the \busobjects folder:
   - add.vbp
   - add.cls
   - add.dll
   - drop.vbp
   - drop.cls
   - transfer.vbp
   - transfer.cls

   All of the above files are Visual Basic 5.0 project files. They have already been compiled into a DLL named **Add** component.
3. Open the State University Web project in Visual InterDev, and add the following files from the \MWD\Labs\Lab09 folder to the root of the Web project:
   - transferclass.asp
   - dropclass.asp
   - addclass.asp

   If copies of these files are already in the project, overwrite the existing versions with the files from \MWD\Labs\Lab09.
4. Add the component add.dll to the State University package by dragging the file from the Windows Explorer, or by clicking **New** from the **File** menu in Microsoft Transaction Server Explorer.

   For more information on adding components, see Adding Components to a Package.
5. In the State University package, set the transaction properties for the **Add** component to **Requires a transaction**. Set the **Activation** property for the **Add** component to **In the creator's process**.
6. Test the **Add** object by opening classview.htm in the browser.

   View the transaction statistics in Microsoft Transaction Server Explorer. Log in to the profile using student ID 1 or any other valid student ID, and try adding yourself to classes.

   If you attempt to add yourself to a class that you are not enrolled in, the transaction should complete successfully and the statistics will show that a transaction completed.

   If you attempt to add yourself to a class that you are already enrolled in, the transaction should fail and the statistics will show that a transaction aborted.
7. Use DataView to verify that the Enrollment table has been updated correctly when classes are added. You can also verify that classes are added by retrieving the transcript.asp file.

In this exercise, you will modify the **Drop** object to call the **SetComplete** or **SetAbort** method. You will also modify the **Transfer** object to use the **CreateInstance** method, so that the object can create the **Add** and **Drop** objects. The **Transfer** object will then call the **Add** and **Drop** objects to perform the work of transferring a student.

Finally, you will add the **Drop** and **Transfer** objects to the new package you created in the previous exercise, and test the package by calling the objects from .asp files.

You will use the transfer.htm file to submit a form that transfers a student. To see an illustration of how transfer.htm looks, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

When you transfer a student using the transfer.htm file, the transferclass.asp file calls the Transfer object to transfer a student. To see an illustration of how transferclass.asp looks when returned to the student, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

u Add the Drop object

1. In Visual Basic 5.0, open the Drop project from the \busobjects folder.
2. On the **Project** menu, click **References**, select **Microsoft Transaction Server 1.0 Type Library**, then click OK.
3. Open the class module drop.cls.
4. At the beginning of the **Drop** function, assign the **Drop** function 0 as a default return value. The return value is used by the .asp file that calls the object to determine if it worked or not.
5. Add code to retrieve the **Context** object from Microsoft Transaction Server.

   To see an example of how your code should look, click this icon.
   {ewc mvimg, mvimage,!code.bmp}
6. Find the call to the **ClassCompleted** function. Add a line of code to call **SetAbort** if **ClassCompleted** is True.

   If the function returns a value of True, it indicates that the student has completed the class, so the **Drop** function aborts.
7. In the error handler, add code to call the **SetAbort** method to indicate that any changes made by the **Drop** function should be undone.
8. After the conn.close statement, add code that calls the **SetComplete** method.

   For more information on the **SetComplete** and **SetAbort** methods, see Adding Transactional Support.
9. Build drop.dll by clicking **Make Drop.dll** from the **File** menu.
10. Use the Microsoft Transaction Server Explorer to add the **Drop** object to the State University package.
11. In the State University package, set the transaction properties for the **Drop** component to **Requires a transaction**.
12. In the State University package, set the **Activation** property for the **Drop** component to **In the creator's process**.
13. Test the **Drop** component by opening the file classview.htm in the browser.

    Log in to the profile using student ID 1 and try dropping the MT100 class. The attempt should fail because the class is already completed. The failure should be logged as an aborted transaction.

    Try dropping other classes. These drops should succeed and be logged as completed transactions.
14. Use DataView to verify that the Enrollment table has been updated correctly when classes are dropped. You can also verify that classes are dropped by retrieving the transcript.asp file.

Note If you drop a class in which the current student ID is not enrolled, the drop will still succeed. This is because the **Drop** method uses the SQL DELETE statement to remove records from the enrollment table. The SQL DELETE statement will succeed even if the record does not exist.

---

u Add transaction support to the Transfer object

1. In Visual Basic 5.0, open the Transfer project from the \busobjects folder.
2. In the project, add the library references **Microsoft Transaction Server 1.0 Type Library**, **State University Add Object**, and **State University Drop Object**.

   Note For their references to be available, the **Add** and **Drop** objects must be registered. To register the objects, add them to the State University package in Microsoft Transaction Server.

   ---

3. Open the class module transfer.cls.
4. At the beginning of the **Transfer** function, assign the **Transfer** function 0 as a default return value. The return value is used by the .asp file that calls the object to determine if it worked or not.
5. Add code to retrieve the **Context** object from Microsoft Transaction Server.
6. Add code to create the **Add** and **Drop** objects by calling the **CreateInstance** method of the **Context** object. Assign the created objects to variables named **addobj** and **dropobj**, respectively.

   To see an example of how your code should look, click this icon
   {ewc mvimg, mvimage,!code.bmp}
7. Before the **Exit Function** call, add code that calls the **SetComplete** method to indicate that the **Transfer** method has completed successfully.
8. In the error handler, add code that calls the **SetAbort** method to indicate that any changes should be undone.
9. To build the Transfer.dll, click **Make Transfer.dll** on the **File** menu.
10. Use Microsoft Transaction Server Explorer to add the **Transfer** object to the State University package.
11. In the State University package, set the transaction properties for the **Transfer** component to **Requires a transaction**. Set the **Activation** property for the **Transfer** component to **In the creator's process**.
12. Test the **Transfer** object by opening transfer.htm in the browser.

    Log in to the profile using student ID 1 and try transferring to a class in which that student is already enrolled. The transfer should fail and be logged as an aborted transaction.

    Then, try transferring between valid classes. The transfer should succeed and be logged as a completed transaction.
13. Use DataView to verify that the Enrollment table has been updated correctly. You can also verify that classes are transferred by retrieving the transcript.asp file.

Article ID: Q164002
Creation Date: 20-FEB-1997
Revision Date: 21-FEB-1997

The information in this article applies to:

® Microsoft Access 97

Symptoms

Moderate: Requires basic macro, coding, and interoperability skills.

When you use Microsoft Internet Explorer to open an ASP file created from a form and a subform in Microsoft Access 97, you may receive the following error message:

> Internet Explorer cannot open the Internet site <URL>. The site was not found. Make sure the address is correct, and try again.

Note that you may receive a different error message using another Web browser.

If you use a proxy server for your Internet access, you may receive an error similar to the following:

> HTTP Proxy reports: The proxy server has encountered an error (Host was not found).

Cause

You did not specify a Server URL (Uniform Resource Locator) when you exported the form to ASP format in Microsoft Access 97.

Resolution

Recreate the ASP files, and specify the correct Server URL in the Microsoft Active Server Pages Output Options dialog box.

More Information

When you export forms to ASP format, Microsoft Access creates several files. The Server URL information is included in the ASP script files that are generated in order to reference the correct path to the subform. The Server URL is the location where the ASP files are stored on your Web server. For example, if you store the ASP files in the \ASPsamp folder on the \\PubTest server, http://pubtest/aspsamp/ is your Server URL. If you do not enter a Server URL when you create the ASP files, your Web browser will be unable to locate and open the subform.

u **Steps to Reproduce Behavior**

1. Create a System DSN on your Web server that points to the Northwind sample database. For more information about creating a System DSN, please see the following article in the Microsoft Knowledge Base:

   ARTICLE-ID: **Q159682**
   TITLE: PRB: "Data Source Name Not Found" Err Msg Opening Web Page
2. Start Microsoft Access 97 and open the sample database Northwind.mdb.
3. Select the Orders form in the Database window, and on the File menu, click Save As/Export.
4. In the Save As dialog box, click "To an External File or Database," and then click OK.
5. In the "Save Form 'Orders' In" dialog box, select Microsoft Active Server Pages (*.asp) in the Save As Type box, and type Orders.asp in the File Name Box. Select a folder to store the ASP files, and then click Export.
6. In the Microsoft Active Server Pages Output Options dialog box, type the name of the System DSN you created in step 1 in the Data Source Name box, and delete any information in the Server URL box. Click OK.
7. Copy the files you just created — Orders.asp, Orders Subform.asp, and Ordersalx.asp — to a folder on your Web server where you have Execute permission.
8. Start your Web browser and type the URL in the address box to view Orders.asp. For example, if you copied your ASP files to the \InetPub\ASPsamp folder on your Web server, type http://<ServerName>/ASPsamp/Orders.asp.

Note that your Web browser returns an error when it tries to open the subform because you did not enter a Server URL in step 6.

References

For more information about exporting ASP files, search the Help Index for "ASP files," or ask the Microsoft Access 97 Office Assistant.

For more information about configuring Microsoft Internet Information Server (IIS) permissions, please refer to the IIS Help Index, and see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q160754**
TITLE: Error "HTTP/1.0 403 Access Forbidden" Browsing IDC Page

Article ID: Q164008
Creation Date: 20-FEB-1997
Revision Date: 21-FEB-1997

The information in this article applies to:

® Microsoft Access 97

Symptoms

Novice: Requires knowledge of the user interface on single-user computers.

When you use Internet Explorer 3.0 or 3.01 on Microsoft Windows NT version 4.0 to browse an Active Server Pages (ASP) file that contains a subform, the subform does not appear.

Cause

Subforms in an ASP file are displayed in a separate instance of Internet Explorer. If Internet Explorer is running Windows NT 4.0, it is not able to display the embedded instance of Internet Explorer.

Resolution

If you run Internet Explorer 3.0 or 3.01 on Microsoft Windows 95, it does not have any problems displaying subforms in ASP files.

Status

Microsoft has confirmed this to be a problem when browsing ASP files in Internet Explorer 3.0 and 3.01 on Microsoft Windows NT version 4.0. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

More Information

Steps to Reproduce Problem

1. Start Microsoft Access 97 and open the sample database Northwind.mdb.
2. Select the Orders form in the Database window, and then click Save As/Export on the File menu.
3. In the Save As dialog box, click "To an External File or Database," and then click OK.
4. In the "Save Form 'Orders' In" dialog box, select Microsoft Active Server Pages (*.asp) in the Save As Type box, and type Orders.asp in the File Name box. Note the folder to which the files will be exported. Click Export.
5. In the Microsoft Active Server Pages Output Options dialog box, enter the name of a System DSN on your Web server that points to the Northwind sample database. In the Server URL box, enter the Uniform Resource Locator (URL) for the location on the Web server where the ASP files will be stored. Click OK.

   For more information on how to define a system DSN, search the Help index for "ODBC, setting up data sources," and see the following article in the Microsoft Knowledge Base:

   ARTICLE-ID: **Q159682**
   TITLE: "Data Source Name Not Found" Err Msg Opening Web Page
6. The ASP output creates three files: Orders.asp, Ordersalx.asp, and Orders Subform.asp. Copy the files to a folder on your Web Server computer where you have Execute permission. This must be the same location you entered in the Server URL in step 5.

   For more information about configuring Microsoft Internet Information Server (IIS) permissions, please refer to the IIS Help Index, and see the following article in the Microsoft Knowledge Base:

   ARTICLE-ID: **Q162975**
   TITLE: Permissions Necessary to View HTML, IDC, and ASP Files
7. Start Microsoft Internet Explorer 3.0 on a computer running Microsoft Windows NT version 4.0.
8. Type the URL in the address box of your Web browser to view Orders.asp. For example, if you saved your

ASP files in a folder called Test in the wwwroot folder of your Web Server, type:

http://<servername>/test/Orders.ASP

Note that the URL depends upon where your files are located on the Web Server.

Note that the Orders.asp form opens and displays in Internet Explorer 3.0, but no subform appears.

References

For more information about exporting forms to ASP, search the Help Index for "ASP files," or ask the Microsoft Access 97 Office Assistant.

Article ID: Q163706
Creation Date: 14-FEB-1997
Revision Date: 17-FEB-1997

The information in this article applies to:

® Microsoft Access 97

Symptoms

Moderate: Requires basic macro, coding, and interoperability skills.

Hyperlink fields in an Active Server Pages (ASP) file created with Microsoft Access 97 are displayed as text with number signs (#) when you view them with a Web browser.

Cause

Data that is stored in a Microsoft Access hyperlink field is stored in three parts (displaytext, address, and subaddress) separated by number signs. This is different from hyperlinks in HTML files, which are created using the <A> tag. Microsoft Access outputs the data in a hyperlink field as it exists, without adding the <A> tags or parsing the data into an HTML format.

Resolution

Create a select query that parses the hyperlink field and adds the appropriate <A> tags. Then export the query to an ASP file, and modify the file so the hyperlink field is not HTML encoded.

If you do not modify the ASP file, the < and > symbols are translated to &lt and &gt respectively, and the hyperlink is still displayed as text.

The following example demonstrates how to create a select query that you can export to ASP format so the hyperlink fields are preserved when you view them in a Web browser. The query contains a column that parses the HomePage Hyperlink field in Northwind's Suppliers table, and adds HTML tags to preserve the links.

Note This example contains information about editing ASP files. It assumes that you are familiar with Active Server, Visual Basic Scripting, and editing HTML files. Microsoft Technical Support engineers do not support modification of any HTML, HTX, IDC, or ASP files.

---

1. Start Microsoft Access 97 and open the sample database Northwind.mdb.
2. Create a new query in Design view based on the Suppliers table.

Note In the field expression below, an underscore (_) at the end of a line is used as a line continuation character. Remove the underscore from the end of each line when you create this example.

```
Query: HyperASP
------------------------------------------------------------------
Type: Select Query
Field: SupplierID
   Table: Suppliers
Field: GoodHyper: "<A HREF=""" & Right([HomePage],Len(IIf(IsNull _
       ([HomePage]),"",[HomePage]))-InStr(IIf(IsNull([HomePage]), _
       "",[HomePage]),"#")) & """>" & IIf(Left([HomePage], _
       IIf(InStr(IIf(IsNull([HomePage]),"",[HomePage]),"#")>1, _
       InStr(IIf(IsNull([HomePage]),"",[HomePage]),"#")-1,0))="", _
       [HomePage],Left([HomePage],IIf(InStr(IIf(IsNull _
       ([HomePage]), "",[HomePage]),"#")>1,InStr(IIf(IsNull _
       ([HomePage]),"",[HomePage]),"#")-1,0))) & "</A>"
```

The expression in the GoodHyper column is so long because it must account for nulls in the hyperlink field, and the NZ function cannot be used with ASP. The expression must also handle hyperlinks that do not contain the displaytext portion of the hyperlink field.

3. Save the HyperASP query and close it.
4. Select the HyperASP query in the Database window, and then on the File menu, click Save As/Export.
5. In the Save As dialog box, click "To an External File or Database," and then click OK.
6. In the "Save Query 'HyperASP' In" dialog box, select Microsoft Active Server Pages (*.asp) in the Save As Type box, and type HyperASP.asp in the File Name box. Select a folder to store the files, and then click Export.
7. In the Microsoft Active Server Pages Output Options dialog box, type the name of a System DSN on your Web server that points to the Northwind sample database in the Data Source Name box.

   For more information on how to create a System DSN, search the Help Index for "ODBC, setting up data sources," and see the following article in the Microsoft Knowledge Base:

   ARTICLE-ID: **Q159682**
   TITLE: "Data Source Name Not Found" Err Msg Opening Web Page
8. In the Server URL box, type a Uniform Resource Locator (URL) of the Web server location where your ASP files will be stored. For example, if you store the ASP files in the \ASPsamp folder on the \\PubTest server, type http://pubtest/aspsamp/ in the Server URL box. Click OK to create the ASP files.
9. Use Notepad or another text editor to open the HyperASP.asp file. Near the bottom of the file you will see the following text:

   ```
   COLOR=#000000><%=Server.HTMLEncode(rs.Fields("GoodHyper").Value)%>
   <BR></FONT></TD>
   ```

   Remove the Server.HTMLEncode portion of the text so it looks like this:

   ```
   COLOR=#000000><%=rs.Fields("GoodHyper").Value%><BR></FONT></TD>
   ```

10. Save the HyperASP.asp file and close it.
11. Copy HyperASP.asp to the folder on your Web server computer that you indicated in step 8. You must have Execute permission in this folder.

    For more information about configuring Microsoft Internet Information Server (IIS) permissions, please refer to the IIS Help Index, and see the following article in the Microsoft Knowledge Base:

    ARTICLE-ID: **Q162975**
    TITLE: Permissions Necessary to View HTML, IDC, and ASP Files
12. Start Microsoft Internet Explorer 3.0, or another Web browser program.
13. Type the URL in the address box of your Web browser to view HyperASP.asp, for example, http://pubtest/aspsamp/HyperASP.asp. Note that the URL depends upon where your files are located on the Web server.
14. Your Web browser returns the SupplierID and the HomePage hyperlink field with the hyperlinks preserved.

More Information

- **Steps to Reproduce Behavior**

1. Start Microsoft Access 97 and open the sample database Northwind.mdb.
2. Select the Suppliers table in the Database window, and then on the File menu, click Save As/Export.
3. In the Save As dialog box, click "To an External File or Database," and then click OK.
4. In the "Save Table 'Suppliers' In" dialog box, select Microsoft Active Server Pages (*.asp) in the Save As Type box, and type Suppliers.asp in the File Name box. Select a folder to store the files, and then click Export.
5. In the Microsoft Active Server Pages Output Options dialog box, type the name of a System DSN on your Web server that points to the Northwind sample database in the Data Source Name box. In the Server URL box, type a URL that points to the Web server location where your ASP files will be stored. Click OK to create the Suppliers.asp file.
6. Copy the Suppliers.asp to a folder on your Web server computer where you have Execute permission.

7. Start Microsoft Internet Explorer 3.0, or another Web browser program.
8. Type the URL in the address box of your Web browser to view Suppliers.asp. For example, if you saved your ASP files in a folder called Test in the wwwroot folder of your Web server, type:

   http://<servername>/test/Suppliers.asp

   Note that the URL depends upon where your files are located on the Web Server.
9. The Web browser returns the HomePage field, but note that the hyperlinks are not preserved, and text containing number signs appears in place of the hyperlinks.

References

For more information about exporting ASP files, search the Help Index for "ASP files," or ask the Microsoft Access 97 Office Assistant.

In a secure environment, the Web server must be able to verify the identity of a client. This process is referred to as client authentication. Microsoft Internet Information Server and Microsoft Internet Explorer can use digital certificates for client authentication.

Client authentication is implemented by the Secure Sockets Layer (SSL) protocol. This protocol requires that both the client and the server provide digital certificates and be authenticated. The focus of this topic is on the client authentication process.

Authentication Process

When a user attempts to access a secure Web page, the following general process occurs:

1. A user requests a secure Web page by using the syntax https:// rather than http://.
2. The Web browser and Web server exchange messages to agree on a hashing algorithm for creating message digests.
3. The Web browser and Web server exchange digital certificates.
4. The client and the server use public key technology to securely exchange messages and agree on a set of random bits that will be used for authentication.
5. The client hashes this set of random bits to create a message digest, uses its private key to encrypt the digest, and sends this to the server.
6. The server uses the public key from the digital certificate provided by the client to decrypt the message received from the client. It then creates another message digest from the set of agreed-upon random bits and compares the two message digests. If they are the same, the client is authentic.

To see an animation on how the authentication process works, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

The complete title of this white paper is *Microsoft Visual Interdev — The Integrated Development Tool For Building Active Server Web Applications.*

This white paper includes the following sections:

While the World Wide Web began its life as a platform for sharing documents over the Internet, today the Web is being used for much more than simple document publishing. In fact, most commercial and corporate Internet sites can be more accurately described as *Web applications* because they require complex processing to create a more compelling, informative experience for users. Web technology is also experiencing widespread growth as an effective platform for deploying corporate *intranet* applications. Unfortunately, however, developers of both public Internet sites and corporate intranet sites are finding that integrated, visual tools for building dynamic Web applications are just emerging, and today's offerings often fail to meet even their most basic needs. While a variety of hypertext markup language (HTML) editing tools are on the market, they are targeted at publishing static Web pages, as opposed to creating and managing dynamic Web applications. These editing tools provide little help for developers building sites that require sophisticated processing, such as server-side components and database connectivity. The lack of integrated Web application development tools is due to the fact that the underlying technology for Web applications is fundamentally different from traditional client-server technology, so traditional rapid application development (RAD) tools do not work for Web applications. As a result, Web developers must resort to a variety of non-integrated tools, often consisting of little more than text editors, complex common gateway interface (CGI) scripting, and various software libraries stitched together in the most rudimentary ways. Besides impacting developer productivity, the lack of integrated, visual development tools is preventing many corporations from more widely adopting Web technology for internal applications, despite the many potential benefits of the technology.

To meet the needs of developers, Microsoft is introducing Microsoft Visual InterDev (formerly code-named "Internet Studio") for building dynamic Web applications for corporate intranets and the Internet. As a member of the Microsoft visual tools family, Visual InterDev was designed from the ground up for developing HTML-based Web applications. In addition, Visual InterDev fully interoperates with the Microsoft FrontPage Web authoring and management tool. Because most Web sites are created by teams of people with different skill sets, the combination of Microsoft Visual InterDev and Microsoft FrontPage offers organizations an effective workgroup development platform for developers and non-programmers alike.

Over the past few years the Internet has blossomed as the most important worldwide communications network, physically connecting consumers and organizations while also providing robust standards for publishing shared information via the World Wide Web. There is no doubt that the Internet has become the de facto *information superhighway*, and Web protocols have become the information publishing standards for this highway. But as the Web continues to evolve from a document publishing platform to a platform for networked applications, a variety of development issues are being introduced.

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. Web Application Components

A Web application can consist of many components. Box 1 in the diagram depicts application logic that runs on the client. On the client, a Web application can include HTML pages with client-side software components such as ActiveX Controls and Java applets running inside the Web browser. The client pages can also include scripting that runs inside the browser — such as VBScript and/or JavaScript. Web applications can also require complex server-side processing. Boxes 2, 3 and 4 depict common server-side components for a Web application. The Web server is responsible for serving-up HTML pages, including dispatching requests to external applications responsible for constructing dynamic HTML pages on-the-fly. For example, Web applications typically call external server applications to process HTML forms into a database server, and to retrieve database records and format them into dynamic HTML pages, which the server then sends to the client for display.

On the client, the Web pages themselves increasingly contain programming logic such as JavaScript or Visual Basic Script, as well as embedded software components such as Java applets and ActiveX Controls that can provide advanced functionality to users. These client-side elements of a Web application are depicted in figure 1, box number 1. Dynamic Web applications must be able to coordinate a variety of components and processing in order to provide user interactivity and up-to-date information (for example, real-time data and dynamic information stored in databases). These Web applications have server-side processing, commonly accomplished with the use of CGI applications, to process forms, respond to user input, and format database information into HTML pages constructed on-the-fly. Often the applications must integrate with existing systems within an organization, such as product and customer databases, as well as order processing and various transaction-oriented systems. The server-side elements of a Web application are depicted in figure 1 as shaded boxes numbered 2-4.

Just as Internet Web sites are evolving into sophisticated Web applications, corporations are also beginning to use Web technology within their private corporate networks, or intranets, as an effective way to build and deploy internal applications. For management information systems (MIS) organizations, intranets provide the following core benefits for internal application development:

- **Decreased deployment costs**. Because intranet applications are server-based, access is seamless for users. The MIS department does not have to distribute client-side software or configure users' desktops. Rather, users can navigate directly to the intranet site and have seamless access to the application with no additional setup or desktop configuration necessary. If the application functionality needs to be changed, MIS professionals can make updates on the server, and all users will instantly be upgraded with the new capabilities. In organizations with thousands of desktops potentially distributed throughout hundreds of remote offices, these benefits alone can justify the use of Web technology for internal application deployment.
- **Cross-platform applications**. Intranet applications are delivered as HTML pages and are cross-platform in nature. This allows organizations with heterogeneous desktop platforms to ensure that all users can access the application without leaving some users out — or requiring special versions for each platform.
- **Low-bandwidth applications**. With Web applications, most processing is done on the server and only HTML pages are delivered to the client (sometimes with small embedded software components and scripting information). Because of this, Web applications are automatically suited for low-bandwidth connections. This is convenient for organizations who have many mobile workers, such as sales agents who need to dial in to access corporate information from the field, or workers who need to dial in to work from their homes.

While these benefits are tangible, however, MIS organizations are constrained because the tools to support Web application development are immature at best — often consisting of little more than a simple text editor!

The lack of tools for developing Web applications is due to the fact that the existing, more mature, client-server tools do not work for Web application development. This is because the underlying network and user interface technologies for Web applications are fundamentally different from traditional client-server applications. For example, traditional client-server development environments can rely on persistent connections between the clients and servers, while Web connections over HTTP (the network protocol used to send and receive information over the Web) are intermittent — constantly being established, broken, and reestablished as the browser requests pages and objects over the network. Also, user interfaces for traditional client-server development make use of multiple windows and are typically based on proprietary forms technologies provided by RAD tools. These tools leverage the underlying operating system to present the various elements of the user interface. A Web-based graphical user interface, on the other hand, is based on a page-browsing metaphor and is constructed using HTML — a document publishing standard that works across platforms. Finally, traditional client-server applications can easily maintain state between the various windows that comprise them (for example, using globally scoped variables); Web applications are a series of loosely connected pages with no notion of persisted state between them.

Because of these fundamental differences, a new category of development tools geared toward Web technologies needs to evolve. To meet this challenge, Microsoft is introducing Microsoft Visual InterDev, a complete development system for building dynamic Web applications for corporate intranets and the Internet. Besides sharing the same look and feel as other Microsoft visual development tools, Visual InterDev makes it easy to integrate components and business processes developed using Visual Basic, Visual C++, Visual FoxPro ,Visual J++, and a variety of third party tools within the context of an overall Internet or intranet solution, thus leveraging an organization’s investment in these tools.

Visual InterDev is a tool specifically designed for developers who want to build sophisticated, dynamic Web applications, as opposed to non-programmers. Visual InterDev supports team-based development and fully interoperates with Microsoft FrontPage 97, a tool targeted at end users and designers. Thus, teams of developers and nonprogrammers can work together on Web sites. Visual InterDev is based on the following key design goals:

1. **Integrated visual development environment.** Visual InterDev includes a comprehensive development environment that integrates all the tools necessary to build and deploy Web applications.
2. **Support for building Active Server Applications.** Active Server Applications are based on *Active Server Pages*, a new feature of Microsoft Internet Information Server 3.0. As a server-side application framework, Active Server Pages make it easy to build dynamic Web applications with sophisticated server-side processing such as database access, state management, server-side scripting and reusable server components. Visual InterDev is the best way to build Active Server applications.
3. **Powerful, integrated database tools.** Visual InterDev offers the most complete and advanced database development features available in any Web development tool. Visual InterDev provides scalable Internet and intranet-based access to any database supporting Open Database Connectivity (ODBC). This includes high-end database management systems such as Microsoft SQL Server, Oracle, Sybase, Informix and IBM DB/2, as well as desktop databases such as Microsoft Access and Microsoft Visual FoxPro .
4. **Site management and content development tools.** Visual InterDev includes complete Web site management that is compatible with Microsoft FrontPage, including a unique site-visualization tool to aid in ongoing site management tasks. In addition, Visual InterDev includes a version of the FrontPage WYSIWYG HTML editor as well as tools for developing Web-based images and sound effects.
5. **Open and extensible.** Visual InterDev is designed to support industry standards such as HTML, HTTP, ODBC, ActiveX, COM and Java. As such, it provides the highest degree of interoperability with other tools and platforms and provides for seamless extensibility through third-party components.

{ewc mvimg, mvimage,!illust.bmp}
Figure 2. Visual InterDev and Microsoft FrontPage

Almost all Web sites are created and maintained by teams of people working together, including both programmers and non-programmers alike. The combination of Microsoft FrontPage and Microsoft Visual InterDev is an ideal workgroup solution, since team members can work on the same Web site simultaneously, each using a tool tailored for his or her specific needs.

This section contains the following topics:

This section contains the following topics:

Visual InterDev includes a variety of visual tools within a single well-integrated development environment. Developers can create and manage shared Web projects, and Visual InterDev will automatically publish the content developed to a Web server and provide ongoing site management capabilities for that Web site. The integrated development environment (IDE) also includes visual database tools for creating and managing data-driven Web sites where Web pages are dynamically constructed based on live connections to databases.

{ewc mvimg, mvimage,!illust.bmp}
Figure 3. Microsoft Visual InterDev

Microsoft Visual InterDev is an integrated development system for building dynamic Web applications for corporate intranets and the Internet. The IDE brings together a variety of visual tools that help increase the productivity of Web developers building sophisticated Web sites. Visual InterDev shares the same IDE as Microsoft Visual J++ and Microsoft Visual C++, enabling developers to have projects using Visual C++, Visual J++ and Visual InterDev open simultaneously within a single, global work space.

The Visual InterDev IDE is based on the next-generation Microsoft Developer Studio shell, first introduced with Microsoft Visual C++ and also used by Microsoft Visual J++. Users of these tools will be immediately familiar with the IDE, and new users will find it very intuitive and easy to learn. It is important to note that the IDE provides an integrated, global work space for Visual InterDev Web sites, Visual J++ projects and Visual C++ projects. Developers can open multiple projects of these types simultaneously within the global work space. For example, a developer might have a Visual J++ applet project open in the IDE as well as an Visual InterDev project so that he or she can both develop and test the applet on the Web site within a single development environment.

An Visual InterDev project consists of a live Web site. Unlike the process involving traditional client-server tools, when developers open a “project” they are actually opening a live view of a site as it exists on the Web server. The server can be a personal development server running on a developer’s workstation, but more typically it will be a staging or production Web server running on a network. The IDE is thus a complete Web site management tool that allows the developer to easily modify the structure of a Web site (for example, adding a subdirectory) and to edit, add, move, rename and delete files and folders on the Web site. Multiple Web sites (projects) can be open at the same time. The Visual InterDev IDE includes a main File View that allows the developer to view and navigate a Web site using the Explorer-based metaphor in the Windows 95 operating system. The Visual InterDev IDE presents the entire Web site, including all content (such as HTML pages, GIF/JPG images, controls and applets, and other files) as well as the complete subdirectory structure of the site.

To edit or examine a particular element of the site, the developer simply clicks the file and it is retrieved from the Web site, copied to the developer’s workstation, and opened in the appropriate editor for that file type. Once the desired changes are made, the developer can save the changes locally, or choose to “release” the file back to the Web server. The project model is multiuser: Multiple developers can work simultaneously on the same Web site. Check-in and check-out capabilities are provided by integration with the Microsoft Visual SourceSafe version control system (a topic covered later in this document). The project system makes use of the FrontPage server-side extensions so that Visual InterDev developers and FrontPage users can work together simultaneously on the same Web sites.

The IDE makes it easy for developers to create new sites with wizard technology and to import content into existing sites, including entire directory trees. Developers are also able to browse pages in the project directly inside the IDE, an important productivity feature, since developers do not have to pull up a separate Web browser window to view a site under construction. The integrated browser is based on the Microsoft Internet Explorer 3.0 Web browser control and supports all its features, including viewing pages with Java Applets, ActiveX Controls, ActiveX Documents, scripting, style sheets and HTML 3.2 features such as tables and frames. The IDE also provides a “preview in browser” feature so the developer can set up other browsers (for example Microsoft Internet Explorer 3.0 and Netscape Navigator) and with a single mouse click preview any page in separate windows running those browsers.

This integration is based on the upcoming release of Visual J++ and Visual C++ 5.0.

Design-time ActiveX Controls are an important new feature introduced with Visual InterDev. They provide all of the benefits of component software offered by standard ActiveX Controls, such as plug-and-play functionality and visual editing at design time. However, design-time ActiveX Controls generate HTML-based content, viewable on any platform and any browser. Design-time ActiveX Controls can also automatically generate server-side or client-side scripting required to accomplish simple or very complex tasks within a Web site. In essence, they are visual helper components that help developers construct dynamic Web applications based on HTML.

It is hard to overstate the importance of design-time ActiveX Controls for Visual InterDev. The product derives maximum extensibility using design-time ActiveX Controls; third-party software vendors and corporations can seamlessly extend the tool with specialized functionality using custom design-time ActiveX Controls. Design-time ActiveX Controls are based on the Component Object Model (COM) and hence can be shared across tools and applications. They can be developed in any language that supports COM such as C, C++, Java, Visual Basic version 5, and many others. Visual InterDev provides several integrated design-time ActiveX Controls, such as the Data Command Control for visually constructing complex data queries and various other data access controls. These controls provide an easy-to-use, visual mechanism for developers to construct complex database connectivity logic within a Web site, with the control doing the hard work of generating the HTML and scripting necessary for the run-time processing of the application.

Microsoft plans to supply a growing set of design-time ActiveX Controls on the Visual InterDev Web site, in effect allowing the live upgrading of the tool with new functionality. In addition, Microsoft expects that over time, independent software vendors (ISVs) will provide hundreds of design-time ActiveX Controls compatible with Visual InterDev, just as there are hundreds of ActiveX Controls available for use with Visual Basic. It should be noted that Visual InterDev also supports standard ActiveX Controls as a way to extend Web pages with client-side software components that execute inside the browser.

Because design-time ActiveX Controls differ from standard ActiveX Controls in that they do not contain a binary, run-time component, Microsoft has decided to name them as a separate category of ActiveX Controls. It is important to note that design-time ActiveX Controls do implement COM interfaces, so they can be arbitrarily shared across development tools provided by various software vendors. However, because they generate HTML and text-based scripting, there is no binary run-time component — hence their output can be viewed on any platform in any browser. These controls are based on published COM interfaces developed during design previews held at Microsoft, and any interested third party can build them. Design-time ActiveX Controls will co-exist with standard ActiveX Controls, which do provide a binary run-time component that offers run-time functionality not offered by design-time ActiveX Controls. For example, standard ActiveX Controls, unlike design-time ActiveX Controls, are exposed as COM objects at run time and design time, and can be scripted at run time by exposed methods, properties and events.

Visual InterDev is an integrated visual development tool that makes it easy to develop dynamic Internet and intranet applications based on Active Server Pages, a new feature for the Microsoft Internet Information Server. As a server environment, Active Server Pages are compatible with the most popular Web browsers, so applications developed with Visual InterDev require no special client software and can be viewed on most browsers on any platform. To understand the flexibility of Visual InterDev to meet the needs of both Internet and intranet developers, it is important to first examine the rich functionality Active Server Pages provide.

Active Server Pages were formerly known as The ActiveX Server Framework. Active Server Pages will be an integrated feature of Microsoft Internet Information Server 3.0. Visual InterDev includes this feature, so that existing users of the Windows NT Server operating system 4.0 and Internet Information Server 2.0 can easily upgrade their Web servers with this new functionality simply by installing the Visual InterDev server components on Windows NT Server 4.0. Active Server Pages can also be used with the development versions of Internet Information Server available on Windows NT Workstation 4.0 and Windows 95. In either case, there are no additional server licensing fees for this feature.

Originally both the Web server and Web client were “dumb,” with a server sending HTML files, which rendered formatted text to a client. This arrangement is still often in use. In this scenario, a user sends a request to a Web server by way of hypertext transport protocol (HTTP) for a particular HTML file. The server receives the request and sends the HTML back to the client’s browser. The browser reads the HTML and displays it accordingly. While this model provides instant access to nicely formatted pages of information for employees or potential customers, interaction between the user and the Web server is limited. In addition, the information is only as up-to-date as the last time someone manually edited the files.

With CGI, ISAPI and other gateway interfaces, a user can send an HTTP request to an executable application instead of requesting a static HTML file. The server immediately runs the specified program. The program can read environment variables and standard input to determine what values were passed with the request (for example, values that a user enters by filling out an HTML form). The program then parses the values for meaningful information and generates output in HTML to send back to the client. The disadvantage of gateway programs is that they are difficult to create and change, and they require an entirely different design process than HTML pages. In addition, CGI-type applications have high server overhead since they require the Web server to change contexts between the server process and the CGI executable process on each program request.

Active Server Pages are based on the ActiveX Scripting engine for Microsoft Internet Information Server 3.0, which allows developers to include server-side executable script directly in HTML content. Hence, applications developed with Visual InterDev are:

® **Content-centric**. Completely integrated with the underlying HTML files

® **Simple**. Easy to create with no compiling or linking of programs

® **Powerful**. Object-oriented and extensible with ActiveX Server components

{ewc mvimg, mvimage,!illust.bmp}
Figure 4. Microsoft Internet Information Server and Active Server Pages

Active Server Pages are a powerful new Web application development framework for building dynamic Web sites. Active Server Pages are HTML pages that contain scripts processed on the server before being sent to the Web browser. These server-side scripts, written in VB Script, JScript, Perl or other scripting language, can make use of custom or packaged ActiveX Server components to extend the Web server with application-specific functionality.

What does this mean for Web application developers? It means the difference between publishing content and providing interactive business applications, as the following examples illustrate.

® For the human resources manager, it is the difference between publishing an HR handbook and providing interactive benefits management.

® For the real estate broker, it is the difference between advertising properties, and providing interactive appointment scheduling, mortgage application processing and more.

® For the travel agent, it is the difference between publishing flight schedules and providing online reservations and ticketing.

® For the mutual fund manager, it is the difference between publishing a prospectus and providing online portfolio management.

® For MIS professionals, it is the difference between setting up a Web site to share documents and setting up a site to provide fully interactive Web applications.

With Active Server Pages, developers can use various scripting languages, including VB Script and JScript to build robust and highly scalable solutions. Visual InterDev provides a complete development system for developing ActiveX Server applications, including a variety of visual tools. Many of these tools automatically generate much of the server scripting necessary to perform complex tasks, such as establishing pooled database connections that can be used globally in a site; performing complex SQL queries against databases and integrating the data into dynamically constructed HTML pages; and creating HTML forms that are directly linked to databases. While the visual tools are powerful and easy to use, Visual InterDev always provides the developer with the option of working directly with source code for maximum flexibility.

The basic ActiveX Server unit is the Active Server Page file, or .ASP file. An .ASP file consists of straight ASCII text in one of three forms:

® Text

® HTML

® Script

Script consists of commands inside ActiveX script tags or HTML <SCRIPT> tags that can be processed on the server by the ActiveX Server engine. When a user requests a URL with an .ASP file extension, the ActiveX Server engine reads through the file from top to bottom, sending text and HTML back to the user's browser, and executing any commands. ActiveX Server scripting is completely integrated with HTML pages. All that's needed to enable the ActiveX Server engine to read a file is to give the file an .ASP file extension. Thus, existing HTML files can simply be renamed from an .HTM extension to an .ASP extension, and they become Active Server Pages. Developers can then make the HTML dynamic by adding ActiveX Server script to the .ASP page. The host language of ActiveX Server scripting is VB Script or JScript. However, developers can use the <SCRIPT> tags to include scripts from other scripting languages, such as Perl.

{ewc mvimg, mvimage,!illust.bmp}
Figure 5. A Simple Active Server Page

An Active Server Page (ASP) is an HTML page that contains server-side scripting. Server-side scripts are designated with an opening and closing <%> tag and are processed before the page is sent to the browser. In this case, a simple loop is performed to display "Hello World" in an increasing font size within an HTML page. The result is shown in Figure 6.

{ewc mvimg, mvimage,!illust.bmp}
Figure 6. Browsing An Active Server Page

Pictured above is the Hello World Active Server Page. All server scripting is processed on the server, and only HTML is sent to the client, as the associated source code viewed from Microsoft Internet Explorer 3.0 shows.

Even though writing ActiveX Server scripts with Visual InterDev is relatively easy, the scripts provide a powerful environment for Internet and intranet applications. For example, developers can incorporate sophisticated functionality through ActiveX Server Components (previously known as OLE Automation Servers). ActiveX Server Components are an extremely important feature of Visual InterDev. ActiveX Server Components do not need to be constrained to the safety restrictions imposed by VB Script and JScript because they execute on a controlled server as opposed to users' desktops. They can thus be used to extend ActiveX Server Scripting with capabilities such as direct access to the file system and access to machine and network resources.

As COM components, they can be driven by VB Script and JScript, and they can execute as either in-process DLLs or out-of-process executables. For lightweight components designed as in-process DLLs, performance can be dramatically improved over CGI solutions, since no context switching between processes is incurred as Web users browse pages that use the component. In addition, ActiveX Server Components can be instantiated once and shared between all users connected to the Web site for more efficient use of server resources. Using Visual InterDev, developers can easily integrate ActiveX Server Components written in Java, Visual Basic (4.0 or higher), C, C++ or other languages that can create COM components.

Visual InterDev with ActiveX Server Components enables developers to easily create multitier Web applications. For example, a component that provides financial modeling and analysis functions could be used to build portions of a financial services Web application. Using Visual InterDev, the developer could create Active Server Pages that execute the component on the Web server, and use the component's methods (functions) to return financial modeling information to users in dynamically constructed HTML pages. Because the component executes on the server, the content can be viewed on any platform by any browser.

ActiveX Server Components provide a convenient and effective way to tightly integrate a Web application with internal and legacy systems. For example, a client-server insurance processing application written in Visual Basic can be exposed as a set of ActiveX Server Components that can be called directly from Web pages by way of ActiveX Server Scripting. In this fashion, Visual InterDev helps to protect and extend organizations' investments in existing tools and systems, ranging from mainframe applications to more recently deployed client-server applications. Visual InterDev serves as the tool that can be used to effectively integrate the functionality of these systems directly into the Web site with ActiveX Server Components.

Because ActiveX Server components are COM objects, out-of-process server components can be seamlessly distributed over a server network using distributed COM (DCOM). This means that components requiring a heavy amount of processing can be distributed to application servers that work in conjunction with the Web server to efficiently process requests from users browsing the site. For example, a price look-up component that performs complex pricing calculations could be built as an ActiveX Server Component and distributed through DCOM to execute on a specialized application server. The advantage of DCOM is that distributed computing is transparent to the developer. The ActiveX Server scripting used to execute and manipulate the remote component is exactly the same as if the component were running directly on the Web server. Distributed solutions built with DCOM components and Visual InterDev offer more effective load balancing, higher performance applications, and greater fault tolerance for enterprise-class Web applications. Active Server Components can optionally be managed using the Microsoft Distributed Transaction Coordinator.

{ewc mvimg, mvimage,!illust.bmp}
Figure 7. Web Applications with Distributed Components

Visual InterDev enables corporations to take advantage of distributed ActiveX Server components with distributed COM (DCOM). DCOM provides complete network transparency, so developers program DCOM components using the same scripting logic as if the components were running locally on the Web server. DCOM components can provide greater scalability and fault tolerance for mission-critical Web applications.

This section contains the following topics:

Powerful database connectivity options and visual database tools are an integral component of Visual InterDev. The database connectivity features are based on the industry standard ODBC, and the visual tools work with any database supporting ODBC, including Oracle, Microsoft SQL Server, Microsoft Access, Microsoft Visual FoxPro, Informix, Sybase, IBM DB/2 and many others. In addition, Visual InterDev enables the creation of scalable database solutions because it leverages Active Server Pages. The core database components of Visual InterDev include these:

® Integrated Data View

® Database design-time ActiveX Controls

® Database wizards

® The Query Designer

® The Database Designer

The integrated database access tools are based on Microsoft Active Data Objects, which provides flexible, object-based database connectivity to ODBC data sources. In addition, Visual InterDev can be used with alternate data access components developed by third parties.

{ewc mvimg, mvimage,!illust.bmp}
Figure 8. Visually Inserting a Data Connection

Visual InterDev provides a visual interface for quickly adding sophisticated database features to a Web site. Pictured above is the data connection dialog box that allows the developer to add a data connection within a Web site to any ODBC data source. The property page allows the developer to visually set properties for the connection, such as the data source, cursor driver, query time-out values and others. Visual InterDev does the hard work of generating the HTML and server-side scripting, automatically saving the logic into an Active Server Page.

Active Data Objects (ADO) are used to provide flexible and scalable database connectivity within Visual InterDev applications. Specifically designed for Web-based data access, ADO provides an object-based approach to data access over the Web. Through ActiveX Scripting, connections to databases can be easily established to any ODBC data source, and a variety of methods within the component provide the developer with a powerful set of database commands for manipulating data and creating data-driven Web pages. ADO allows manipulation of database-defined data types, including binary large objects (BLOBs) such as GIF and JPG images retrieved from databases and dynamically written into Web pages. In addition, ADO provides a rich set of properties for setting locking levels, cursor options, query and login time-outs, transaction support, result set scrolling, error handling and more. Using VB Script or JScript, ADO provides developers with maximum flexibility to develop powerful database functionality within their Web sites.

Visual InterDev includes the Data View feature, which provides a visual interface to all of the databases being used within a Web site. Besides depicting each database connection being used in the site, the Data View provides a live connection to each database, allowing developers to work directly with these databases within the IDE during Web site development. For example, the developer can open any database to view tables, defined views and even stored procedures. The Data View can provide detailed information on objects and properties within each database, including table definitions and field types, key structures, stored procedures and so on. The Data View works with the Query Designer and Database Designer features and provides a sophisticated database development, administration and maintenance system tightly integrated with the Visual InterDev IDE. The Data View works against any ODBC-compliant database and will show multiple connections against heterogeneous databases.

{ewc mvimg, mvimage,!illust.bmp}
Figure 9. The Visual InterDev Data View

Visual InterDev includes an integrated view into any ODBC data source being used within the Web site. Pictured above, the Data View pane on the left shows an open data source to a Microsoft Access database and a Microsoft SQL Server database, including its tables, fields and views for each.

Visual InterDev builds on the ADO foundation by providing special design-time ActiveX Controls that automatically generate much of the server-side scripting, including the ADO calls necessary to establish database connections within a Web site, perform queries and display results. In many cases, the developer can use the design-time ActiveX Controls to create data-driven Web sites with little additional programming required. Visual InterDev, however, provides developers with the ability to view and develop directly in ActiveX Server scripting using the ADO component for maximum flexibility.

One example of a design-time control is the Microsoft Data Command Control. With the Data Command Control, the developer can select a database connection to an ODBC data source and then visually build a query against that connection using the Visual InterDev Query Designer. Once tested and complete, the Data Command Control generates all of the ActiveX Server scripting necessary to perform the query and automatically adds the scripting information in the appropriate Active Server Page. Data range controls provide similar functionality for visually building complex data queries, but also generate the looping and display logic necessary to page through a result set as a series of dynamically constructed HTML pages.

In addition to design-time ActiveX Controls, Visual InterDev also provides wizards that lead the developer through the process of creating custom data-bound HTML forms. After prompting the developer for information, the Data Form Wizard will automatically generate the HTML and ActiveX Server scripting information required to create complex HTML forms that are bound to databases. For example, a developer could use the Data Form Wizard to visually construct a guest book form, complete with the logic to accept user input, update a database of registered users, and display a listing of all users registered in the guest book database. Because the wizards, like design-time ActiveX Controls, generate standard HTML and ActiveX Server scripting, developers can modify the generated source code as desired for further customization.

{ewc mvimg, mvimage,!illust.bmp}
Figure 10. The Data Form Wizard

The Data Form Wizard steps the developer through the process of building sophisticated and highly customized HTML forms bound to databases. The wizard automatically generates the HTML and server-side scripting logic, which can be further modified, if desired, by the developer.

In addition to providing the visual Query Designer, which works against any ODBC-compliant database, Visual InterDev also provides 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 create new SQL Server databases and modify the structure and properties of existing SQL Server databases. Plus, database administration operations that used to take hours can be accomplished with a couple of clicks of the mouse. For example, a developer or database administrator can use the Database Designer to change the data type on a SQL field (e.g., from a type CHAR to a type INT) with a simple drop-down selection. Visual InterDev will then automatically change the field type, which ordinarily would require 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. The Database Designer can generate scripts of DDL that can be reviewed and submitted to database administrators for review and execution in controlled database environments.

The Database Designer can also be used to set up complex database designs with unique and check constraints and to define relationships between tables using foreign keys. By allowing developers to functionally group tables together, database diagrams can be dragged into the Query Designer to quickly construct queries against these logical groupings.

{ewc mvimg, mvimage,!illust.bmp}
Figure 12. The Database Designer

The Database Designer allows developers to create database schemas for Microsoft SQL Server databases. The tool provides a powerful and flexible database administration environment that simplifies the most complex SQL Server administration tasks.

Some database tools for Web applications offer visual aids for easing the development of data-driven Web sites but do not provide the scalability required for Internet sites or production intranet sites hosting mission-critical applications. Visual InterDev, however, provides both ease of use with extensive visual tools, and the scalability required for mission-critical, data-driven Web sites. For example, global database connections can be established for an entire site, and Internet Information Server will automatically pool these database connections across users. Pooling, connection caching and time-out values are all established automatically based on default properties, but they can easily be customized by the developer. Visual InterDev also makes it easy to connect to multiple heterogeneous databases within a Web site, returning or updating data that has been visually integrated within a single HTML page for the user. For example, Visual InterDev can be used to return data from an Oracle database running on a UNIX server, as well as data from a SQL Server database running on a Windows NT-based computer, all integrated within a single HTML page for the user.

This section includes the following topics:

Visual InterDev includes extensive site management features integrated within the development environment. These features begin with the integrated project system that presents a live view of the site as it exists on the Web server. The project system provides the developer with the ability to create new Web sites, view content on an existing Web site, and to create, add or modify folders and files within a Web site. Specific site management features within the project system include these:

- **Creation of new Web sites with Web site wizards.** The wizards step the user through the process of creating new Web sites, automatically creating a new site on the Web server chosen.
- **Direct file manipulation of any item in a Web site.** Developers can copy, delete, add, rename and edit any file in the Web site from within the IDE, using the File View feature.
- **Ability to import content files and directory trees into a Web site.** Developers can use the Import File and Import Folder commands to import existing content from any local or network resource into the Web site.
- **Copy Web command.** An entire Web can be moved from one server to another server using the Copy Web command. This is useful for moving a Web from a staging server to a production server, for example.
- **Object properties.** Developers can right-click on any object on the Web site, and Visual InterDev will present a property page that contains useful information, such as file size, modification date and all incoming and outgoing links to a file.

The project system is able to track links within a Web site and ensure they are valid. For example, should a developer rename a file, the system will automatically detect all the pages throughout the site that reference that name as part of a URL reference (hyperlink) and automatically fix these references to avoid broken links. In addition, should a file be moved or the site physically restructured using Visual InterDev, the system will automatically repair links in this manner. Finally, should a page or file be deleted, Visual InterDev will notify the user of all the references to that file that exist within the site.

Besides providing developers with automatic link repair and site management capabilities through the File View, Visual InterDev also has a unique Web visualization tool, called Link View. Link View presents a graphical view of the Web site, visually showing the logical relationship of pages within the site. For example, Link View can be used to visually depict the links from a particular HTML page to other pages inside or outside the Web site. Developers can expand or collapse the view to any level, showing as many pages or sections of the Web site as they want. Besides diagramming links, Link View also depicts specific information on each file, using visual icons to depict its file type, such as HTML pages, Active Server Pages, GIF/JPG images, ActiveX Controls, Java Applets, 2.5D Layout files and even design-time controls. Link View will graphically depict all broken links in red, as well as show external links in addition to internal links. Link View thus provides a smart design-time view of the site, and developers can use the graphical view to launch editors on each file simply by double-clicking the file.

With Link View, the developer is also able to filter the view so that only specific information is depicted. A developer may choose to filter the view to show (in any combination):

® HTML pages

® Multimedia files

® Executables

® Documents

® External files

® Primary links

® Secondary links

{ewc mvimg, mvimage,!illust.bmp}
Figure 13. LInk View

Link View allows developers to visualize the logical structure of a Web site. Pictured above, Link View shows a graphical representation of two HTML pages within a Web site and the references these pages have to various elements, such as GIF images and WAV files. Broken links are depicted in red.

Visual InterDev provides integrated source management for users of Microsoft Visual SourceSafe 5.0. Organizations using Visual SourceSafe can place a Web project under source control. Source management is especially helpful when teams of people will be working on the same Web site, since it provides explicit check-in and check-out capabilities for all files in the site. In addition, the Visual SourceSafe revision tracking and merging features help developers to protect their work — giving them, for example, the ability to roll back to an earlier version of a file or to compare in detail the differences between two versions of a file.

Visual InterDev integrates with Visual SourceSafe so that any project can easily be put under source control without requiring Visual SourceSafe itself to be installed on the developer's workstation. Instead, all Visual SourceSafe features execute on the server. For Visual InterDev projects under source control, the Visual SourceSafe integration features include the following:

- **Automatic creation of Visual SourceSafe projects.** When a project is placed under source control, Visual InterDev will automatically create the Visual SourceSafe project on the Web server, and add the existing content of the Web site to the source control database.
- **Check-in and check-out files.** As developers request to edit individual files in the Web site, Visual InterDev will explicitly check the file out of the Visual SourceSafe project before opening the file. Should another person already have it checked out (for instance, another Visual InterDev user or a FrontPage user working on the same project), the developer will be notified and permitted a read-only copy.
- **Automatic addition of elements to a Visual SourceSafe project.** As developers create new files for a Web site, Visual InterDev will automatically put these elements under source control, using the Visual SourceSafe project set up for the site. For example, if a developer chooses to import a folder that contains 10 files and two subfolders, Visual InterDev will automatically add these elements to the Visual SourceSafe project. Should a user create a new file, such as a new HTML page, Visual InterDev will create the new entry in the Visual SourceSafe project and automatically check the element out to the developer for editing.

Besides offering an integrated, visual environment for developing the next generation of server-based Web applications and data-driven Web sites, Visual InterDev also provides integrated tools for developing Web pages, including Web pages with client-side scripting, ActiveX components and Java Applets. In addition, Visual InterDev includes multimedia content editing tools: Microsoft Image Composer and Microsoft Music Producer. These tools allow developers to easily develop sounds and images for use in a Web site.

Visual InterDev includes a version of the award-winning FrontPage 97 HTML editor. Thus, teams of developers and business end users can share a common, leading-edge HTML editing environment while working in the tools specifically designed to meet their differing needs.

Visual InterDev includes integrated support for adding ActiveX Controls, Java Applets and Netscape plug-ins to Web pages. When working with ActiveX Controls, developers are presented with a visual Object Editor that allows them to visually modify the component to fit their needs. A visual property table, similar to the Visual Basic property table, allows developers to easily modify a component's properties without ever needing to see or work with source code. After visually working with a component, the Object Editor will automatically generate all of the HTML syntax (based on the W3C industry-standard OBJECT tag) to bring the component into the Web page and execute it. Perhaps most important, the ActiveX Controls are displayed in true WYSIWYG fashion directly in the FrontPage 97 editor, and can even be dragged, dropped and sized directly within the page. With over 1,000 ActiveX Controls currently available, providing a wide range of functionality ranging from multimedia to multipoint conferencing to real-time data feeds, Visual InterDev offers true extensibility.

The integrated FrontPage editor also provides support for easily inserting Java Applets and Netscape plug-ins. Developers can visually set characteristics such as size and position, and the editor will display the frame for the Java Applet or Netscape plug-in within the document.

Visual InterDev also includes a client-side Script Wizard, first introduced with the ActiveX Control Pad. The Script Wizard makes it easy to add interactivity to Web pages, based on actions and events associated with ActiveX Controls. The Script Wizard includes a visual interface for easily connecting an event (such as a mouse click) with an action (such as playing a video clip). The Script Wizard lists all of the events associated with a given ActiveX Control in the Event Pane. The developer can select an event in the Event Pane and associate it with an action in the Action Pane. Multiple actions can be added in this manner to a single event, without requiring any programming, since the Script Wizard generates the actual lines of script code added to the HTML page.

{ewc mvimg, mvimage,!illust.bmp}
Figure 14. The Script Wizard

The Script Wizard provides a visual interface for building client-side scripts that wire together various ActiveX Controls on a Web page. The Event Pane allows the developer to select from a list of events and wire an event to an action in the Action Pane. The Script Wizard generates both VB Script and JScript code and provides a code entry window where developers can conveniently enter lines of custom code.

Using the Script Wizard, developers can easily integrate the behaviors of multiple controls, thus creating an integrated application from potentially diverse sets of components, with each component providing a specific functionality. The Script Wizard will generate either VB Script or JScript code, depending on developer preference. While aiding the developer in connecting events to client-side scripts, the Script Wizard's code-entry window always permits the developer to code in straight VB Script or JScript if desired, or to modify the code generated. All client-side scripts developed with the Script Wizard are embedded as text within the HTML document using the W3C HTML <`Script`> tag.

Visual InterDev also includes a 2.5D (or “frame-based) HTML Layout Editor introduced with the Microsoft ActiveX Control Pad. The HTML Layout Editor, similar to a Visual Basic form, makes it possible to precisely place multiple ActiveX Controls, providing the developer with exact control over x, y coordinate placement and z-ordering (layering) of controls. But rather than converting the form-based layout information into a best possible representation in HTML (which currently provides no 2.5D layout capabilities), the HTML Layout Editor saves the information using new (draft) W3C syntax that extends HTML with 2.5D layout capabilities. This provides developers with exact control over object placement and object layering on a Web page. The HTML Layout regions use the new style sheet attributes specified in a W3C draft specification for Cascading Style Sheets (see http://www.w3.org/pub/WWW/TR/WD-layout.html) and are rendered by the HTML Layout Control, which is part of the Microsoft Internet Explorer 3.0 Web browser.

HTML Layout regions provide developers with a form upon which they can exactly position any number of ActiveX Controls. Scripting within HTML Layout regions is also possible by using the Script Wizard. The HTML Layout Editor thus makes it possible to build rich, interactive user interfaces for Web pages. As the W3C finalizes the specification for 2.5D layout in HTML, support for browsing 2.5D regions will become native to Microsoft Internet Explorer and quite likely to other Web browsers. Please refer to the Microsoft HTML Layout Control white paper (http://www.microsoft.com/workshop/author/layout) for more technical information on Microsoft’s implementation of 2.5D-style layout for HTML.

{ewc mvimg, mvimage,!illust.bmp}
Figure 15. The HTML Layout Editor

Visual InterDev includes a 2.5D HTML Layout Editor, similar to a Visual Basic form. The Layout Editor provides developers with exact control over the placement of objects on a page, including x, y, and z coordinate control. Instead of converting the form information into a best-possible HTML representation, the layout information is saved using new W3C syntax that extends HTML with 2.5D layout capabilities. The layout information is rendered in the HTML Layout Control, which ships with Microsoft Internet Explorer 3.0.

The HTML Layout Editor offers the following editing features:

- **Toolbox control customization.** Right-clicking on the toolbox allows users to add and delete individual controls supplied by third-party vendors.
- **Toolbox custom tabs.** New toolbox tabs can be added using the toolbox shortcut menu.
- **Toolbox object templates.** Users can drag objects from the form back to the toolbox, creating an object template. The template captures all of the properties set by the user and can be used to easily incorporate objects with custom properties into new pages. Groups of controls can also be used as object templates in this manner.
- **Multiple level undo and redo.** Most edit operations can be undone easily.
- **Control alignment, spacing and sizing.** Controls can be aligned, sized and spaced automatically for easier layout.
- **Control drag-and-drop.** Controls can be positioned exactly with drag-and-drop editing. Also, controls can be dragged from one page to another page.
- **Control layering (z-ordering).** Object layering can be set by right-clicking on any control.
- **Script Wizard.** The Script Wizard can easily be invoked from the menu to add interactivity to a 2.5D layout region.

The HTML Layout Control provides the run-time rendering of 2.5D layout regions within Microsoft Internet Explorer 3.0. It is included in the “Complete” installation of Microsoft Internet Explorer 3.0, and is also available for separate, automatic download as an ActiveX component. The HTML Layout Control also works in Netscape Navigator on Windows 95 and Windows NT, through the ActiveX plug-in for Netscape.

Microsoft Image Composer is especially useful for developers who need to modify or create images to fit a Web site, and it is designed as an easy-to-use tool for developers who are not graphics professionals. It will recognize most popular image formats, including directly reading Adobe Photoshop files, while making it easy to save images into GIF or JPG formats for use on the Web. It is easy to arrange, customize and create images in Microsoft Image Composer for use on Web sites developed and maintained in Microsoft Visual InterDev. Users can launch Microsoft Image Composer from Visual InterDev by clicking a toolbar icon or clicking images directly from within the IDE. Microsoft Image Composer has a command that automatically saves images directly to the Microsoft Visual InterDev project.

For detailed information on this tool, click here:
{ewc mvimg, mvimage,!intjump.bmp}

In addition to Microsoft Image Composer, Visual InterDev also includes a unique music creation tool for creating sound effects for use on Web sites. Music Producer employs the pioneering Microsoft Music Technology (MMT), and simplifies music creation with prebuilt, innovative musical styles. More versatile and powerful than static music clips, styles define entire musical genres (such as samba or Texas swing) as well as musical moods and sets of instruments. Over 100 predefined styles are included. Even first-time users can create music immediately because everything needed to create quality original music is presented in one simple screen.

Microsoft Media Manager is a utility that makes it easy for developers to track and manage dynamic and static media assets such as images, sound clips, video clips, HTML files and office documents. It increases productivity by allowing developers to quickly search, preview, play and retrieve media assets stored in Media Manager libraries. The utility integrates seamlessly with the Windows 95 and Windows NT operating system and uses the same explorer-based navigation paradigm. Microsoft Media Manager provides extended search capabilities, thumbnail previews of images, and previews of other media content such as sound clips and video clips directly from the library navigation (explorer) pane. Media assets can be simply dragged and dropped from a media library into an Visual InterDev project.

The open and extensible architecture of Visual InterDev is a critical aspect of the tool, since it provides a very high degree of flexibility for developers. Visual InterDev delivers on the promise of openness by doing the following:

® Supporting the HTML and HTTP Internet standards

® Supporting ODBC for connectivity to all popular database systems

® Delivering “thin-client” cross-browser and cross-platform Web applications

® Providing full client-side and server-side extensibility via ActiveX components and Java Applets

® Supporting VB Script and JScript client-side and server-side scripting, as well as other scripting engines through third-party scripting engines

Visual InterDev is an integrated, visual development tool designed to meet the needs of developers who want to build dynamic database-driven Web applications for corporate intranets and the Internet. Because Visual InterDev fully interoperates with Microsoft FrontPage, a tool designed for end users and graphic designers, teams of developers and non-programmers can work together on Web sites. Visual InterDev delivers on the following key design goals:

® Integrated, visual development environment

® Support for building Active Server Applications

® Powerful, integrated database tools

® Integrated site management and content development tools

® Open and extensible

Visual InterDev is a comprehensive Web development system, including integrated programming, database development, site management, and content editing tools. Developers will find that Visual InterDev helps increase their productivity by providing a single, integrated development environment that includes easy-to-use, visual tools for building sophisticated Web applications. By using the powerful database development tools in Visual InterDev, programmers can work with any database that supports ODBC. Finally, Visual InterDev delivers HTML-based Web applications that are browser- and platform-independent, while also providing open extensibility with third-party components that can be used to seamlessly extend the tool's capabilities. Corporations will find that Visual InterDev provides them with a distinct business advantage for their Internet and intranet-based applications.

You can also drag files and folders from Windows Explorer directly into a Web project, or into a folder within a project. This will publish the files, over HTTP, to the appropriate location on the Web server.

Click here to launch the ActiveX Control Pad Setup program.
{ewc mvimg, mvimage,!exec.bmp}

The ActiveX Control Pad is an authoring tool that lets you add ActiveX controls and ActiveX scripting (Visual Basic Scripting Edition (VBScript) or JScript) to your HTML pages with point-and-click ease.

The ActiveX Control Pad also includes the new Microsoft HTML Layout Control, which provides exciting new layout capabilities based on the new HTML extensions proposed by the World Wide Web Consortium (W3C). Using the ActiveX Control Pad, you can easily author pages that include advanced layout and multimedia features such as exact object placement, object layering, and transparency effects.

System Requirements

To run Microsoft ActiveX Control Pad 1.0, you need the following:

® A personal computer with a minimum 80486 microprocessor running Microsoft Windows 95 or Windows NT 4.0

® 12 MB of RAM

® 10 MB of free disk space on your hard drive

® The latest release of Internet Explorer 3.x

Before installing Microsoft ActiveX Control Pad, make sure that you have the latest release of Microsoft Internet Explorer 3.x installed on your machine.

For More Information

After running Setup, open the ActiveX Control Pad README file to review important information about the product.

Note This tool is in the \Tools\ActX_CP directory on the *Mastering Web Site Development CD-ROM*.

Click here to launch the Advanced Data Connector program.
{ewc mvimg, mvimage,!exec.bmp}

Microsoft Advanced Data Connector is a high-performance Web-based technology that brings plug-and-play database connectivity and corporate data publishing capabilities to Internet and intranet applications.

With Microsoft Advanced Data Connector (ADC), you can build intelligent Web applications that let you access and update data from an ODBC-compliant Database Management System (DBMS). And because you implement ADC with familiar technology — off-the-shelf visual controls, HTML, and Microsoft Visual Basic Scripting Edition (VBScript) — ADC integrates seamlessly with existing Visual Basic applications, letting you transport them to the Web. Because they are based on familiar technologies, ADC applications provide the added benefit of leveraging the existing knowledge base of Visual Basic developers, thereby decreasing development time.

Microsoft Advanced Data Connector uses Open Database Connectivity (ODBC), the standardized architecture that supports building data-aware applications without targeting a specific DBMS. Using ODBC, you can easily modify ADC applications to suit different network and DBMS configurations.

In addition to ODBC, Advanced Data Connector is integrated with other Microsoft technologies, including Internet Information Server (IIS) 3.0, Microsoft Transaction Server, and OLE DB.

A key feature of Microsoft Advanced Data Connector is a client-side caching mechanism that minimizes connections to your DBMS. As a result, you get better performance than in traditional Web database access methods. Such performance improvements are especially noticeable when accessing data across the Internet.

Setup instructions

The setup program for the Advanced Data Connector product is msadc10.exe. It should be installed on the Web server computer which would be running NT Server 4.0 or later. This provides the documentation and some sample files. Setup assumes that the Software Requirements have already been installed.

Setup installs the program files to the <device>:\Program Files\Common Files\System\MSADC directory. This location is adjacent to other complementary Microsoft technologies such as ActiveX Data Objects and OLE DB.

The msadc10.cab file has been provided to simplify the deployment of ADC enabled web pages by taking advantage of HTML's automatic code download feature. Automatic code download is enabled by specifying the CAB file path and name in a CODEBASE parameter when creating an AdvancedDataControl object. When the msadc10.cab file is downloaded it automatically installs the required client files.

For More Information

For further information and details about Advanced Data Connector, check out the Readme file, or click here to visit the ADC Home page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Note This program is in the \Tools\ADC directory on the *Mastering Web Site Development CD-ROM*.

Click here to launch the License Package Authoring Tool.
{ewc mvimg, mvimage,!exec.bmp}

An HTML Page with licensed controls requires a single associated license package, which stores the run-time licenses for all the controls used on the page. The HTML page should point to a license package via a relative URL. The License Package Authoring Tool (LPK) allows you to author a License Package (LPK) file.

Using LPK_TOOL

LPK_TOOL displays two list boxes:

® **Available Controls List Box**

This box lists all controls registered in your system.

® **Contorls in License Package List Box**

This box lists all controls, whose licensing information, if available, will be stored in the LPK file. The Licensing information will be save when you invoke the 'Save & Exit' button.

Add Button

You can select 'controls' in the 'Available Control List Box' and use the 'Add' button to instruct the tool to add it to the LPK file. LPK_TOOL responds by moving the selected controls to the "Controls in License Package" List Box.

Remove Button

You can select 'controls' in the 'Controls in License Package List Box' and use the 'Remove' button to instruct the tool not to add it to the LPK file. The LPK_TOOL responds by moving the selected controls to the 'Available Control List Box'.

Save & Exit Button

The 'Save & Exit' button instructs the tool to save the licensing information to the LPK file. LPK_TOOL prompts for a file name and saves the Licensing information in the file name provided. LPK_TOOL terminates after saving the file.

Cancel Button

The 'Cancel Button' terminates the application without creating the LPK file.

About Button

The 'About Button' displays the version information.

Help Button

The 'Help Button' dislays the help file for the tool.

Show only Controls that support Licensing Check Box

When checked, LPK_TOOL displays only those controls, which support Licenses (IClassFactory2) interface.

For More Information

For more information about the tool, refer to the LPK_Tool.HLP file. For more information on Licensing, please refer to the technical white paper, Licensing ActiveX Controls.

Note This tool is in the \Tools\LPK_Tool directory on the *Mastering Web Site Development CD-ROM*.

The LPK_TOOL makes use of the License Manager components. Please make sure the License Manager is installed on your system and is registered before you use this tool. (The License Manager is installed with Internet Explorer.)

Click here to launch the self-extracting file for the Microsoft Script Debugger program.
{ewc mvimg, mvimage,!exec.bmp}

Microsoft Script Debugger is a debugging environment that extends Internet Explorer 3.01 by allowing Web developers to browse, edit, and debug HTML pages that contain Visual Basic Scripting Edition (VBScript) or JScript (Microsoft's implementation of JavaScript).

System requirements

Before you install this release, make sure that you have:

® An x86 processor running Windows 95 or Windows NT 4.0

® Internet Explorer version 3.01 installed on your computer

® 10 MB of free disk space •At least 16 MB of RAM

Note This release does not provide international or DBCS support, and does not run on RISC platforms. This product has not been localized and will NOT work on non-English versions of Microsoft Internet Explorer.

Starting Microsoft Script Debugger

The Microsoft Script Debugger Setup program installs the debugger into the same directory as Internet Explorer. Once you have installed Microsoft Script Debugger, you can start it using one of the three methods below (you must restart Microsoft Internet Explorer first):

® Choose Source from the View menu in Internet Explorer. Script Debugger starts, and opens the current HTML source file. This is the only way to start the debugger outside of a script. (If the Script Debugger is not installed, clicking Source starts the Notepad editor.)

® Choose Break at Next Statement from the Edit menu and then execute a script. The debugger window opens, and stops at the first statement in the current script.

® Execute a Stop statement (VBScript) or debugger statement (JScript) in a script. Script Debugger opens, and stops at the statement.

To switch back to your original version of Internet Explorer (without the debugging features enabled), choose the Add/Remove Programs option from the Control Panel, and select the Script Debugger from the list of installed programs (it will be listed as "Microsoft Script Toolkit for Internet Explorer").

For More Information

For more detailed information on using Microsoft Script Debugger, see the help file included with the tool (choose Contents from the Script Debugger window Help menu), or click here to connect to the Script Debugger Web page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Note This program is in the \Tools\ScriptDb directory on the *Mastering Web Site Development CD-ROM*.

Click here to launch the Visual Basic Script Documentation installation program.
{ewc mvimg, mvimage,!exec.bmp}

This executable will install VBScript HTM files. There are several ways you can view the VBScript documentation. You can double-click the icon that Setup creates. You can also start your browser, click Open on the File menu and type the file name, VBSTOC.HTM, into the dialogue box.

VBSDOC.exe contains the following materials:

® Language and run time reference

® Using VBScript in HTML

For More Information

For more information about VBScript, click here to connect to the Visual Basic Script Web page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Note This program is in the \Tools\VBSDoc directory on the *Mastering Web Site Development CD-ROM*.

Visual InterDev provides a sophisticated SQL Query Designer that works against any ODBC data source. The integrated Query Designer provides an extremely easy-to-use interface for visually constructing even the most complex SQL statements, and generates the Data Manipulation Language (DML) for SELECT, INSERT, UPDATE and DELETE queries. Developers can open live views of data sources and drag tables directly into a design pane of the Query Designer window to build their queries. As the developer selects fields from tables, the SQL pane will show the dynamically constructed SQL statement. Developers can directly modify the SQL statement, and the changes will be reflected in the query design pane. In addition, the Query Designer allows developers to execute any SQL statement to test it and displays the results in a results pane. The Query Designer enables developers to easily create 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, execute arbitrary database definition language (DDL) commands directly against any ODBC data source, or perform ad hoc SQL queries. Visual InterDev thus provides a complete, tightly integrated database development and administration tool for Web developers. It is important to note that the Query Designer works in conjunction with the Query Control (a design-time control), so that once developed and tested in the Query Designer, the HTML and server scripting necessary to execute the query is automatically generated and embedded in the appropriate Active Server Page.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11. The Query Designer

The Visual InterDev Query Designer allows the developer to visually construct complex SQL queries against any ODBC data source.

Click here to run the Internet Information Server Fix self-extracting zip program.
{ewc mvimg, mvimage,!exec.bmp}

The iisfix.exe is automatically executed with the -D command, so the directory structure will properly be rebuilt on your system. The iisfix.exe file contains the following files for the i386 and Alpha platforms:

® Q163485.txt

® Q164059.txt

® Readme.txt

® infocomm.dll

® ssinc.dll

® w3svc.dll

® Debug Symbols

After the files are downloaded to your computer, move the files appropriate to your platform and implement as stated below. The symbols directory contains the DEBUG symbols for each platform and these files need not be copied, unless you have been advised to do so by a Microsoft Support Engineer.

Hotfix Installation Instructions

This hotfix requires the replacement of the following files:

® Infocomm.dll

® ssinc.dll

® w3svc.dll

These files are located in the \winnt\system32\inetsrv directory.

u **To implement the fix**

1. Stop all services in the Internet Service Manager.
2. In \winnt\system32\Inetsrv directory, rename or remove...
   - Infocomm.dll
   - ssinc.dll
   - w3svc.dll

   Note ssinc.dll and w3svc.dll are interdependent files and should only be replaced as a pair.
3. Restart the services in Internet Service Manager.

The behaviors described in KB Articles Q163485: Active Server Pages Script Appears in Browser and Q164059: Execution File Text Can Be Viewed in Client should no longer exist. If the behaviors do persist, reboot the server.

Note This program is in the \Tools\IISFix directory on the *Mastering Web Site Development CD-ROM*.

You can also enter a relative URL in the **From Location** box. For example, you could enter logo.gif in the **From File** box. If you have a working copy of logo.gif on your machine, the image will appear on the page as you edit it. If you don’t have a working copy of logo.gif, you can still view the image when you preview the page in the browser.

When you design your Web site, you can use a service-based application model. The term service-based means that the functionality of an application is specified as collections of services that meet specific user needs.

There are three categories of services that make up an application:

® User services

® Business services

® Data services

User Services

User services provide the application with its interface. The user of a service can be a person or another service. Therefore, the interface for a service can provide a graphical user interface and/or a programmatic interface.

For example, Microsoft Internet Explorer has a rich graphical user interface that enables a user to browse a variety of Web sites. Along with its graphical user interface, Microsoft Internet Explorer provides a programmatic interface that offers the same features and functionality with an Automation server.

Business Services

Business services enforce business rules and handle transactions. These services change data into information by using the appropriate business rules.

For example, a business rule can define the minimum set of data required to create a new customer in the corporate database.

Data Services

Data services provide low-level manipulation of data in a database. They provide services, such as create, read, update, and delete, that are used by business services to modify a database. A business service does not need to know where data is located, how it is implemented, or how it is accessed. These tasks are handled by data services.

Examples of data services include Microsoft SQL Server and various file-based storage technologies, as well as multimedia data such as images and video.

To see an animation that introduces the service-based application model used by a Web site, click this icon.
{ewc mvimg, mvimage,!anim.bmp}

Certificate Authorities use public key pair technology to guarantee the identity of the digital certificate owner.

How Public Keys Work

Digital certificates are based on public key pairs, which is an encryption technique using two related keys (a public and a private key). Each of the keys in the key pair is the inverse of the other key, and each performs a one-way transformation of the data it is applied to. What one key does, only the other key can undo. The owner of the key pair makes the public key available to everyone and keeps the private key secret.

There are two ways data can be encrypted with public key pairs:

- A holder of the public key can encrypt data, and only the holder of the private key can decrypt the data. As long as the private key has not been compromised, public key pairs provide a safe means for ensuring one-way private communication to the private key holder.
- The holder of the private key can use the private key to encrypt data. Any owner of the public key can decrypt the data and be guaranteed which private key was used for the encryption.

Public Keys on Digital Certificates

Each digital certificate contains the public key from a public key pair that has been associated with an identity by the CA.

Anyone who trusts that the CA verified the identity of the requester can be assured of the identity associated with the public key pair. Without the verification efforts of the CA, anyone generating a public key pair could claim to be someone else. Digital certificates only guarantee identity to the extent that you trust the CA.

Remember that profile.asp is in the StateU virtual directory.

Use the **InStr** function to compare the saved session variable with the string "profile."

To edit the file global.asa, you must restart the WWW service of your Web server because Visual InterDev uses HTTP to save the files on the server. This will reset the hit counter to zero.

If the hit counter file does not exist, calls to the **OpenTextFile** and **ReadLine** methods will fail, and the hit counter will be reset to zero.

To use the constants **ceCertPresent** and **ceUnrecognizedIssuer**, you must include the client-certificate include file in your ASP page.

If you are using VBScript, include Cervbs.inc. If you are using JScript, include Cerjavas.inc. These files are installed in the \Inetpub\ASPSamp\Samples directory.

Because the format of digital certificates is public, anyone can create them. In addition, the digital certificates themselves are made public by the issuing CA. It is possible to obtain and alter a digital certificate. To be completely secure, it is important to be able to confirm the authenticity of a digital certificate.

What Determines Authenticity?

A digital certificate is considered authentic if the following two points are true:

® The digital certificate has not been altered since it was issued.

® The digital certificate is guaranteed to have been issued by a trusted CA.

For information on how to specify which CAs you trust, see Understanding Digital Certificates in this chapter.

Proving Authenticity

To enable a Web server to verify the authenticity of a digital certificate, a CA includes a digital signature as part of the digital certificate.

u **To create a digital signature, a CA performs the following steps:**

1. A one-way hashing algorithm is applied to the data found in the certificate. The result of the one-way hash is called a message digest.
2. The message digest is encrypted with the private key of the CA who issued the certificate. Only the message digest is encrypted, not the digital certificate itself.

When a Web server receives a digital certificate, it can use the following process to determine if it is authentic.

1. It retrieves the CA name from the digital certificate to determine if the digital certificate was issued by a trusted CA.

   If the certificate was not issued by a trusted CA, it is rejected immediately.
2. It then uses the public key of the issuing CA to decrypt the digital signature and obtain the message digest.
3. It creates a new message digest by applying a hashing algorithm to the digital certificate. It uses the same hashing algorithm that was used by the issuing CA.
4. It compares the new message digest with the message digest from the digital signature.

   If they are the same, the certificate is authentic and is accepted.

   If they are not the same, either the message was signed by another private key or it was altered. In either case, the certificate is rejected.

By Mitchell I. Kramer
November 1996
Published by the Patricia Seybold Group

The complete title of this article is *Microsoft Transaction Server: A General Purpose Infrastructure for Multitier Applications*

This white paper includes the following sections:

Scalability and the Internet Are the Driving Forces

Not much more than a year ago, multitier applications were at the leading edge of development. Only a few tools supported this architecture, and only a small number of production-level applications implemented it. Today, multitier applications are mainstream. Two factors are responsible for this dramatic change: scalability and the Internet. Two-tier applications don't scale well, and developers ran out of memory and processing resources on client platforms. As application logic grew in size, they needed some place to deploy the code. Finding places to run code other than on the client is also a key element of the Internet factor. For Internet applications, logic must be deployed on the middle tier. Everyone wants to Internet-enable existing applications or begin to do "e-commerce on the Web." With the requirements to support thin clients, all these Internet applications must have multitier applications.

Transaction Server Simplifies Middle-Tier Development

The Microsoft Transaction Server is designed to simplify the development of the middle tier of multitier applications by providing much of the infrastructure needed to execute the business logic that runs there. It will insulate developers from dealing with such issues as connectivity, security, directory, process and thread management, and database connection management — the infrastructure elements required to support robust, production-class, run-the-business Internet, intranet, or internal LAN applications.

Thirty to Forty Percent Reduction in Development

Microsoft estimates that Transaction Server will reduce the development effort for multitier applications by between 30 and 40 percent. We agree with that estimate, as do several developers with whom we spoke. A case study presented in this report identifies and qualifies the areas where Transaction Server delivers these reductions.

This Report

This report describes and analyzes the advantages of Transaction Server compared with custom approaches to creating the infrastructure needed to support multitier applications. It is divided into four sections. The first section will present the requirements for this infrastructure. In the next, the issues that developers have identified in building such an infrastructure will be examined. In the third section, we'll examine a real-life example of a custom-developed infrastructure: a customer contact system built by a line-of-business development team at a major bank in the Northeast. The final section will describe the capabilities of Transaction Server and analyze how Transaction Server might have been used for the customer contact system.

Two-Tier Applications Were Easy

In order to get some perspective on the magnitude and complexity of the effort to build multitier applications, contrasting them with earlier two-tier structures will be useful.

The Database Did All the Work

The databases at the back ends of two-tier client/server applications insulated developers from most systems, from networking, and from management coding. Databases performed connectivity, security, directory, transaction processing, and recovery. In addition, database routing software on the client handled all communications. As a result, the databases freed developers to concentrate on the business aspects of the application.

Databases also gave these applications performance and scalability. The number of users that an application could support was a function of the database. Developers ran on clients. Their code supported a single user. This code could be written inefficiently or might run on client platforms with limited resources, limiting performance and scalability, but improving performance and, especially, scaling up were database related.

Two Factors Expand the Scope of Development

Developers could concentrate on application logic when they built two-tier applications, but multitier applications are forcing them to expand development efforts to include infrastructure as well. Connectivity and the processing environment on the middle tier will require the most work. Two factors are responsible for the increased development burden:

- ® The user interface and the application logic are on separate platforms.
- ® Application logic is shared among all users.

Addressing these factors will complicate application development. Developers have told us that that 30 to 40 percent of the total time taken to build applications can be spent on these infrastructure tasks. Illustration 1 shows the structure, components, and architecture of three-tier applications.

{ewc mvimg, mvimage,!illust.bmp}
Illustration 1.

The components of three-tier applications are clients, application servers, and databases. Clients at tier one provide the user interface and, optionally, some portion of the application's logic. Application servers at the middle tier implement most of the application's logic. Databases at tier three provide persistent storage. Clients connect to application servers through a program-to-program protocol. Application servers connect to databases via SQL.

The User Interface and the Application Logic Are On Separate Platforms

In two-tier architectures, all network communication between clients and servers uses SQL-related syntax. Either clients send SQL statements to the database, or the database sends status codes and data back to the client. In three-tier architectures, this program-to-data communication takes place between tiers two and three. Program-to-program communication is needed between clients on tier one and servers on tier two in order to connect application logic that runs on both tiers in multitier environments.

Connectivity Must Be Provided

To accomplish this communication, developers must write the code to connect clients with application servers, transfer data between them, and synchronize and coordinate their processing. Data flows must conform to some protocol in order for the communication to have any meaning. That protocol must define the format and structure of the data transferred, the meaning of their content, the sequencing and timing of sending and receiving, and the definition of errors and the approaches to be taken to recover from them.

Application Logic is Shared Among All Users

In multitier architectures, application logic is shared among all users from a server. This is the biggest difference

between two- and three-tier approaches, and the one that has the biggest impact on application development. The middle-tier environment must provide security, directory, performance and scalability, and availability and reliability, to the application. It must also provide database access and transaction management.

Security

The application servers that run on tier two implement applications' business logic and access databases in response to client requests. Because they may perform sensitive processing and update sensitive data, access to them must be secured. Ideally, a client must be authenticated, authorized, and granted access privileges before an application server processes a client's request. Security is very important in internal corporate environments; it is critical for Internet applications.

Directory

Directory capabilities are closely linked to security. A directory should contain information about all clients, servers, and databases for an application environment. This information should include identifiers, network locations, passwords, authorizations, and access privileges. It should be accessible to each client and server to assist in satisfying requests for services from the perspectives of location resolution and security.

Performance and Scalability

Performance and scalability are key aspects of the middle-tier environment. Ideally, to address performance requirements, each client request should be serviced on demand. In order to address scalability requirements, developers should be able to plan on having the ability to service an ever-increasing number of client requests. The middle tier should support unlimited on-demand concurrency.

Availability and Reliability

Application servers must be available to provide services whenever clients request them. When an application server is down, so is the business that it supports. However, servers can be unavailable for many reasons. Some are:

- ® Server platform hardware problems
- ® Server platform software problems
- ® Networking problems
- ® Application server problems

The code for the application server and the code for the environment that supports its execution must be bulletproof. Its reliability must approach that of the operating system on which it runs. Obviously, it must be thoroughly tested.

Recovery and Restart

When failures do occur, application servers must have the capability to quickly recover from those failures and restart themselves. Recovery should recapture as much user work as possible, and data must be left in a consistent state as well. If a server platform will be down for an extended time, the application server might be restarted on a different platform.

Transaction Processing

Transaction processing is another aspect of processing that was handled by the database in two-tier environments. In cases where multiple, heterogeneous databases were accessed, developers used transaction processing monitors. In multitier environments, a transaction monitor approach will be required, driven by access to multiple heterogeneous databases, mainframe integration, and supporting resource managers other than databases — message queues, for example.

Increased Cycle Time

It takes longer to develop three-tier applications because there is more to build in them than there is in two-tier applications. And this is no small difference. Our experiences and the experiences of developers with whom we've spoken have consistently shown that 30 to 40 percent of a three-tier project can be spent on infrastructure items.

Complexity

Not only does it take longer to build three-tier applications, but the additional development tasks are considerably more complex. In addition to the business domain aspects at the core of applications, developers get involved in security, directory, performance and scalability, availability and reliability, and recovery and restart. These are some of the most sophisticated aspects of systems and networking. These areas had been the domain of so-called systems programmers.

Skills

Developers of two-tier applications were skilled in their business domain and in the use of their selected development tools. Multitier applications will require that development organizations increase their skills base considerably through training, hiring, or consulting services. The technologies for which these additional skills are needed have long, steep learning curves. Hands-on experience has been a crucial factor in building these skills.

Systems Software Built and Maintained by Corporate Developers

Once a multitier application has been deployed, developers have the ongoing responsibility of maintaining, extending, and enhancing both the application's business logic and its infrastructure components. Developers must perform these life-cycle activities not only in response to changes in the business but also in response to changes in operating systems, networking protocols, development tools, and databases. Infrastructure components are "closer" to systems than business logic is. Although they might be built on standard and/or open APIs, these APIs are subject to change, especially in the Internet environment, and frequent system changes can cause problems in their proper operation. As a result, maintenance activities can be complex and time-consuming, and they can occur far more frequently than business changes.

Stovepipes and Silos

One of the most troublesome aspects of successful multitier development projects is that their success is project oriented. All of their infrastructure components were designed and built to address requirements for an individual project. They can't be used for other applications because they're too closely aligned with the application for which they were built. And, more than likely, they're entangled with business logic and can't be identified and extracted as modular components. They are stovepipes or silos — not sharable, not extensible, and not reusable.

This section incudes the following topics:

A line-of-business development team at a major Northeastern bank built a three-tier customer contact system. Requirements for performance, scalability, availability, and ease of maintenance drove the architecture to three tier. This was a very successful project, especially since it was the bank's first three-tier experience. A large team of developers from the bank and from a national consultancy completed the project in 18 months. The system supports over 900 users with very good performance. In this section, the system and the developers approach to the three-tier infrastructure are described. Illustration 2 shows the components, structure, and architecture of this application.

Tier One

In tier one, clients were built with Visual Basic. They make up the user interface, minimal application logic, and the connectivity to tier-two application servers. Dynamic link libraries (DLLs) accessed via program calls provide the connectivity. Clients also provide computer-telephone integration (CTI).

Tier Two

Application servers at tier two provide the major portion of the application's business logic. They were built in C and deployed across four Unix platforms. The code was written procedurally, and it incorporates all application capabilities through a single entry point.

Tier Three

Tier three is a single Informix RDBMS. The database, which is dedicated to the customer contact application, is deployed on one of the four server platforms. In addition to the RDBMS, some tier-three processing involves access to mainframe DB2 databases through Customer Interface Control System (CICS) transactions.

{ewc mvimg, mvimage,!illust.bmp}
Illustration 2. Structure, Components, and Architecture of the Customer Contact System

The customer contact system was a three-tier client server application. Clients at tier one were built in Visual Basic. They provided the system's user interface. Application servers at the middle tier implemented all of the application's logic. Client-to-application server connectivity was accomplished through a custom-built protocol on top of TCP/IP sockets. Two databases made up tier three, an Informix RDBMS located on one of the four application server platforms accessed through SQL, and a DB2 RDBMS accessed through CICS transaction programs located on a remote mainframe accessed through a custom-built SNA LU6.2 gateway.

Connectivity

In an approach quite common for initial three-tier projects, the development team used Transmission Control Protocol/Internet Protocol (TCP/IP) sockets as the connectivity mechanism. The reasons were that sockets were available on all client and server platforms, they are well-understood, and they offer good performance.

Connectivity between the tier-two server and the mainframe was provided through a custom Systems Network Architecture (SNA) gateway using the LU6.2 CPI-C protocol, one of the native CICS protocols. As a result, no coding was required on the mainframe, but building the gateway was a major undertaking.

Considerable additional function was built on top of the sockets interface and the Common Programming Interface-Communication (CPI-C) gateway. For the sockets API, a message structure and protocol were created, a time-out capability was added, and a message-queuing capability was built. The application server wrote responses intended for clients to a persistent queue if clients were unavailable. For the SNA gateway, the development team also built time-out and message-queuing capabilities. A fixed number of SNA sessions was bound between the application server and the mainframe. When all sessions were in use, the next request was queued.

Security

No security capabilities were built into the customer contact system. Rather, the development team leveraged existing security. Users log into Unix on the server platforms with user IDs and passwords. The user IDs are then authenticated through Remote Access Control Facility (RACF) on the mainframe using the SNA gateway to transfer this information. Once users were authenticated, no additional security processing was performed.

Directory

Only rudimentary directory capabilities were developed. A configuration file on each client platform contained the network address of each of the four servers. Users attempt to log into the first server on the list. The four entries in the list provide the only reliability and recovery in the system. That is, when a server is unavailable, the user logs into the next server.

Performance and Scalability

The development team invested significantly in the performance and scalability of the customer contact system. Message queuing, process management, and a performance management system were built.

The performance management system was a major undertaking. The development team instrumented the application in order to collect information in real time at many key points during application processing. The information was managed in a memory buffer and displayed on X-Window terminals attached to the server platforms. Operators and administrators got up-to-the-minute information on response times and queue depths. This information was used for load-balancing and capacity-planning.

An instance of the application server was launched in an operating system process for each client request up to a maximum number of concurrent threads. When the maximum number was reached, subsequent messages would be queued until a process completed.

Transaction Processing

The capabilities of the Informix RDBMS and of CICS were used to provide transaction processing to the customer contact system. No attempt was made to coordinate Informix transactions with CICS transactions, although this distributed transaction processing support will become a requirement in the future.

Recovery and Restart

The Informix RDBMS provided most of the recovery and restart function in the system through its transaction commit/rollback and its journal-based roll-forward and rollback capabilities. No recovery/restart capability was provided for tier-one clients. At tier two, messages representing responses to client are persistently queued when the client becomes unavailable, and requests for mainframe services, RACF authentication, or CICS database access are queued when all SNA gateway sessions are busy.

Availability

Availability was delivered through the ability of any client to log into any of the four servers. As described in the paragraph on directory above, clients maintain a configuration file of all four server addresses. When one server fails, they can log into another.

The development team leader estimated that 40 percent of the development effort was devoted to building the described infrastructure. The infrastructure addresses the needs of the customer contact system, but, the team leader admitted that it cannot be used for other applications and it would be difficult to modify or extend it to meet future application requirements.

Note that the infrastructure provides little or no capabilities in the areas of security and directory. Database pooling is not provided, either. Any efforts in these areas would have driven up the infrastructure development effort to half or more of the total project. On the other hand, the SNA gateway and the performance management system will not be common custom-built infrastructure capabilities.

A Corporate Charter to Build a Common Infrastructure

Corporate IT at the bank has recognized the magnitude of the infrastructure development effort and the fact that it cannot be reused to support other applications. As a result, IT is leading a new effort to define requirements for an infrastructure and to address those requirements with open, standards-based products. The goals are to provide an infrastructure on which all future applications can be built and to minimize infrastructure aspects of future application development efforts. At this writing, requirements have been defined, and the bank is in the process of product evaluation.

This section includes the following topics:

® A General Purpose Infrastructure
® Connectivity
® Security
® Directory Services
® Performance and Scalability
® Reliability and Availability
® Transaction Processing
® Global Information Sharing

The Microsoft Transaction Server is an infrastructure product that can address a broad range of requirements in developing and deploying multitier applications. In fact, it would be a very good fit for the distributed processing requirements at the bank. Developers shouldn't be confused by the name of this product. Transaction Server can free developers from most infrastructure coding in addition to transaction processing functions. Let's take a closer look at what it does and how developers can take advantage of its capabilities, keeping in mind the development efforts at the bank described above and the more general issues of building multitier applications discussed earlier.

Three-tier Structure and Components

Transaction Server supports the multitier application structure and components that we've been discussing throughout this report. Clients at tier one provide the user interface and some amount of the application logic. Application servers at the middle tier provide most of the application logic. They run under the control of Transaction Server. Databases at the back end provide persistent storage for application information.

Component-based Applications and Application Development

Transaction Server assumes component-based applications and a component-based approach to development. In general, we feel that components offer developers the best way to build distributed applications. Their object-oriented properties of modularity, information-hiding, and inheritance make them easy to extend and easy to maintain. Because they're large and coarsely grained, they very naturally implement entire business entities such as customers, products, or employees. So, at development time, they're easily reused, and, at deployment time, they simplify application partitioning and distribution.

COM and Activex

Transaction Server leverages Microsoft's Common Object Model (COM) and ActiveX technology. COM provides the connectivity and defines the interfaces and protocols for communication within and between application components. Developers build and implement application components as ActiveX modules. The most salient description of Transaction Server is that it's a complete tier-two server environment for ActiveX.

As we mentioned in the last paragraph, connectivity in the Transaction Server environment is provided by COM. When a user or a program references the interface of an ActiveX component by sending it a message or an event, COM will locate the component and route the message or event to it. The referenced component may be running on a local or remote platform. Local connectivity is provided by inter-process communications capabilities of Windows 95 or Windows NT. Remote connectivity is provided by TCP/IP networking.

In this environment, developers have to do little or no connectivity programming. They have only to make the ActiveX reference in their code. COM and DCOM (the distributed, networking extension to COM) do the rest. Directory services, described below, provide the key information to COM and DCOM for resolving ActiveX references.

Transaction Server leverages the security capabilities of Windows NT. Windows NT provides platform-based login via user ID and password and access control to resources by users and groups. Transaction Server extends access control to the individual interfaces of application components. This fine-grained access control may be implemented administratively within Windows NT users and groups. It may also be implemented programmatically within application components.

Transaction Server's security capabilities can minimize or eliminate many security-related developments. Its administrative approach will be adequate for many applications. If developers choose a programmatic approach, then they have the advantage of building on an existing structure. Transaction Server does not integrate with the security capabilities of non-Windows systems, nor does it provide rigorous authentication capabilities. Requirements in these areas would have to be addressed through programming.

Transaction Server has several built-in capabilities to help developers address requirements for performance and scalability.

Automatic thread and process management

Transaction Server manages a thread pool for application server components. In response to a request for an ActiveX service, it allocates a thread, calls the ActiveX component on that thread, and returns the thread to the pool when the call completes execution. The assignment of the ActiveX components to server processes is determined by settings in the directory. This allows the components to be easily grouped to meet security and fault-isolation requirements, and it is especially attractive when developers integrate new components into a production environment.

This feature can reduce development significantly while improving performance and scalability. Thread and process management is an area that must be addressed for application servers. Here, developers get into the guts of the operating system. It's easier to work with processes, but it's more efficient to work with threads. Threads require less processing and memory overhead, but they're harder to manage. Custom-built approaches typically deal with processes, trading performance and scalability for ease of development. Transaction Server completely insulates developers from threads and processes.

Database Connection Pool

The database connection pool establishes and manages a set of sessions between the Transaction Server environment and databases that will be accessed by application server components. Rather than establishing database connections inline with application processing either on a request-by-request basis or on a session basis for each client, Transaction Server offers pools of pre-connected sessions. The processing overhead incurred when connections are created and terminated occurs all at once within Transaction Server initialization and termination. That way, these expensive operations are removed from normal request execution, improving application performance. Memory usage is optimized because the size of the pool can be much smaller than what is required by statically assigning a connection to each client.

The database connection pool can reduce development effort and can improve application performance and scalability. Developers are completely insulated from the efforts to establish and manage database connections. They simply code the SQL statement to access information in their applications. Database connection pooling is a well-proven and frequently used technique. The development team in our banking example had planned to build a database connection pool but did not because of schedule constraints.

DCOM Support

Developers can build highly scalable applications through DCOM support in Transaction Server. Application components may be distributed across several server platforms and may communicate with each other through DCOM. Developers access remote components in exactly the same manner as they access local components. DCOM does the work of locating components through the Windows Registry, where components register themselves when they are installed.

DCOM may also be used to perform load-balancing within multiserver environments. Transaction Server does not build in load-balancing capabilities, however. Developers would have to implement this capability themselves.

Transaction Server does not provide capabilities that explicitly address requirements for reliability and availability other than its support for transaction processing. Failover, the routing of a service request to an alternate server platform when the primary platform is unavailable, may be implemented by developers. They can leverage DCOM support to reduce the task of developing this important facility.

Transaction Server provides a transaction monitor that controls transactional access to resource managers. Transactions may access a single resource manager, or, through support of Microsoft's Distributed Transaction Coordinator (DTC) protocol, transactions may coordinate and synchronize access to multiple resource managers. Resource managers in the initial release of Transaction Server will be Microsoft SQL Server and those databases that support the ODBC interface. Future releases will implement XA, SNA LU6.2, and TIP (Transaction Internet Protocol) protocols and, therefore, expand the types of resource managers and processing environments that can participate in transactions.

No Coding Required

The development effort to implement transactions is minimal. Developers do not have to specify Begin transaction, End transaction, or Abort transaction statements in their code. Rather, transactions are implemented by setting a property of an application component. If the component is marked "transactional," then Transaction Server will build a transaction around its processing. And any components referenced by the transactional component automatically participate in the transaction as well.

Transaction Server will affect the design of application components. If components are transactional, then all of their processing will be encapsulated in a single transaction. Developers must take care to keep the scope of processing granular with respect to resource manager access.

The Shared Property Manager of Transaction Server provides a mechanism to share and access global information among server processes. Developers can use the Shared Property Manager for a range of tasks, including workflow, performance monitoring and management, and process serialization through global locks. Implementation of the performance management system in our banking example would have been facilitated by the Shared Property Manager capability.

The Shared Property Manager eliminates the housekeeping tasks needed to share information among applications. It takes care of memory management, freeing developers to concentrate on program logic. Also, because it's a centralized facility, the fragmentation and leakage that might occur when individual developers create and manage global data pools for single purposes can be eliminated.

This section includes the following topics:

Middle-tier application components that run under control of Transaction Manager are implemented as ActiveX DLLs. Developers may build these components in many languages using a range of development tools. Supported development tools currently include:

® Café from Symantec

® Delphi from Borland

® PowerBuilder from Powersoft

® Visual Basic from Microsoft

® Visual C++ from Microsoft

® Visual J++ from Microsoft

® Visual Object Cobol from MicroFocus

Language Independence

The range of tools supported for ActiveX development makes ActiveX and, in turn, the Transaction Server environment virtually language independent. Language independence means that Transaction Server will be easy to learn and easy to use. The ActiveX modules deployed as middle-tier server components are easy to create. Developers have to remember not to include visual components. The hard part is in designing them to be modular.

Developing Transaction Server Components

Transaction Server is not completely transparent to developers, but it comes very close. To build a Transaction Server component, developers will make use of one new API call and one new object interface. The API call, GetObjectContext, provides developers with the information needed to understand the processing state and context of a call to a server component. The object interface, IObjectContext, provides several methods for managing this state of a server component, including performing programmatic access control and indicating the outcome of the call.

Invoking an Application Server Component

To invoke a Transaction Server component, clients or other Transaction Server components create an instance of that component. In Visual Basic, this is done with the CreateObject statement. In Visual C++, developers use the CoCreateInstance API. The parameters of these statements specify the component name and an object reference. COM, the Windows Registry, and the Transaction Server work together to locate the component and create a new instance (object) for the client. This is how Visual Basic Remote Automation servers and other remote OLE servers had been invoked prior to ActiveX and Transaction Server. There is nothing really new here, except that the Transaction Server also establishes and manages the execution context for the object. For example, this is how transactions are automatically provided from components.

Transaction Server components are deployed as *packages*. Packages are the unit of deployment and distribution for Transaction Server applications. Their definitions have considerable flexibility. Examples are:

- The components for an entire application may comprise a single package that is deployed on a single server.
- The components for an entire application may be partitioned among several packages are deployed on a single server.
- The components for an entire application may be partitioned among several packages that are deployed across multiple servers.

Administrators use the Transaction Server Explorer, a visual administrative tool, to partition Transaction Server applications and to create packages. There is no developer involvement in packaging and partitioning tasks.

Reduced Tme and Skills

Microsoft Transaction Server provides infrastructure facilities that can significantly reduce the time and skills required to develop and deploy multitier Internet, intranet, and internal LAN applications. This product insulates developers from complex systems-oriented tasks, such as process and thread management. It also eliminates the need for them to get involved in tedious, time-consuming tasks such as directory management. Combined with the capabilities of DCOM, Transaction Server offers a very attractive package.

An Estimated Two-Thirds Reduction in Infrastructure Development at the Bank

If the Transaction Server could have been used for the custom contact system in our banking case study, we estimate that approximately two-thirds of the 40 percent of the development effort devoted to infrastructure could have been eliminated. Transaction Server would have reduced infrastructure development in several areas, such as the following:

- ® DCOM would have provided most of the connectivity functionality that the development team built on top of sockets. Developers would not have had to write any communications code.
- ® Process and thread management capabilities would have simplified development, and, through the use of threads instead of processes, both application performance and scalability could have been improved. Transaction Server would have eliminated all development done for the customer contact system in this area and would have provided a better solution.
- ® Transaction Server could have provided the database connection pool that the development team did not have the time to build, thus improving performance and scalability.
- ® Shared Property Manager would have reduced the effort to build the performance management system. Memory management coding could have been eliminated. However, most of the performance management system would still have had to be developed.
- ® SNA LU6.2 resource manager support, available in a future Transaction Server release, could have simplified integration with RACF and CICS. Developers would not have had to custom-build an LU6.2 gateway, which is a very complex task.

Some Infrastructure Development Would Still Be Required

Some aspects of the customer contact infrastructure are not addressed by Transaction Server. Those are persistent queuing, load-balancing, and failover, although the bank's static client configuration files can hardly be considered to be a robust implementation of load-balancing or failover.

Reusable for Future Applications

By and large, however, the advantages of Transaction Server compared with custom infrastructure development are quite significant. And, from a broader perspective than the development of a single application, the Transaction Server infrastructure is not a stovepipe solution. Its infrastructure is sharable among multiple applications, and components built to run in its environment may be easily reused.

A Good Start, With More to Come

The infrastructure built into the Transaction Server environment can have an immediate, positive impact on the development of multitier, component-based applications. At first release, it will have a distinctly Microsoft orientation. Indeed, DCOM is its foundation. But future releases will begin to offer good integration with non-Microsoft resources. For example, support for XA and SNA LU6.2 will provide access to a broad range of resource managers and will integrate mainframe resources. In addition, through Microsoft partnerships with Digital and Software AG, DCOM will be supported on many non-Microsoft platforms.

Directory services are provided to Transaction Server and COM through the Windows Registry, Microsoft's directory for Windows 95 and Windows NT. The registry contains information about users, groups, applications, and ActiveX names and interfaces. It provides all the capabilities required to support multitier applications in the Microsoft Windows environments. Creating and maintaining the registry is an administrative task. Developers are insulated from it.

To obtain a client digital certificate from a CA and install the certificate in Microsoft Internet Explorer, you follow these general steps:

1. Access the Web site of the CA.
2. The Web page downloads the certificate enrollment ActiveX control, which creates a certificate request and submits the request to the CA.
3. The CA creates the digital certificate and notifies you when it is available.
4. Once the certificate is available, you access the CA Web site again to install the certificate into Internet Explorer.

To request a digital certificate from the Verisign Certificate Authority, go to their Web site by clicking this icon.
{ewc mvimg, mvimage,!intjump.bmp}

u **To see the certificates installed in Microsoft Internet Explorer**

1. On the **View** menu, click **Options**.
2. Click the **Security** tab.
3. Click **Personal**.

   A dialog box that contains a list of the client digital certificates that have been installed is displayed. To see an illustration of this dialog box, click this icon.
   {ewc mvimg, mvimage,!illust.bmp}

Once your digital certificate is installed, Microsoft Internet Explorer automatically sends it to a Web server when you visit sites that request a digital certificate.

The process to obtain and install a server certificate is somewhat different from installing a client certificate. For details on how to install a server certificate, see the IIS documentation.

Contents of a Digital Certificate

When you request a digital certificate, you supply information that uniquely identifies you. The certificate authority uses this information and creates a digital certificate.

A digital certificate includes the following information:

® Owner information

General information such as name and e-mail address and a public key that is used when you submit a digital certificate. For more information on public keys, see Associating Identity with a Certificate in this chapter.

® Name of the issuing CA

® Digital signature of the issuing CA

The digital signature verifies that the CA issued the certificate. It is also used to verify that the certificate has not been altered since it was issued.

By Louis Kahn and Laura Logan
Briefly described in Books Available from Microsoft Press
Published by Microsoft Press

The following chapters have been included from *Build Your Own Web Site*:

® Chapter 4: Under Construction

® Chapter 5: Hands Off My Hard Disk

Most domains on the Internet are on a lower level of a hierarchy of domains. For example, in the domain *guiware.com*, *guiware* is a second-level domain and *com* is the top-level domain. In order to have a domain on the Internet, you must register it with the naming authority that is responsible for the top-level domain where your domain name will be included. An organization called the *InterNIC* is the naming authority for the top-level domains *gov* (government), *edu* (education), *com* (commercial), *org* (organization), and *net* (for ISPs). As you probably noticed, the top-level domains classify different kinds of organizations. Most of the domains have strict qualifications for you to meet before you can be approved for that domain. Most people fall into the *com* domain because there are no qualifications to meet to be included in this domain, but you still need to decide which top-level domain you properly belong in.

In this chapter, we discuss the process of getting a domain in the *com* domain. If you think that your organization belongs in a different top-level domain and would like to see the qualifications for domains other than *com*, you should consult the InterNIC at *http://www.internic.net*.

Once you have decided which of the top-level domains you belong in, you should choose a name for your own domain. This can be more challenging than it sounds. You need to choose a name that has meaning for you, serves as a marker for your Internet site, is easy to remember and type, and is not already in use by another organization on the Internet. If you have a trademark on a name and find that someone is already using it as a domain name on the Internet, you must go through legal channels to get the name released to you if that organization does not want to give it up.

After you have picked an available name, you need to register the name. The most common way to do this is to connect to the InterNIC's Web page at *http://www.internic.net*. Go to the Registration area, and choose either to download a text-based template file or to use a WWW form. If you choose the text-based template file, you will fill in the required information using a text editor or a word processor. If you use the WWW form, you answer the same questions, but you fill in the blanks using your Web browser software.

The InterNIC needs to know the following details: first, who will be the contact for your domain? The organization recognizes three official contacts — an administrative, a technical, and a billing contact. These can be different people or all the same person. Second, you need to supply the InterNIC with names and IP addresses of the DNS servers that will maintain your domain. The InterNIC requires that registered domains have at least two DNS servers. If you are handling your own DNS (which we don't recommend), then you should be able to supply that information yourself. Otherwise, your ISP will give you the DNS address information. Finally, the InterNIC wants to know how you want to be billed. The current fee registration for a domain name is $100.00. Half of this fee is a one-time setup fee, and the other half is the first payment of a recurring maintenance charge payable every two years to keep the domain name.

E-mail the completed form to the InterNIC at the address *hostmaster@internic.net*. Typically, you receive two e-mail notices from them, the first being a confirmation that they received your request, and the second being the approval of your domain name. See Figure 4-1 below.

Click here for more information on how to register a domain name and to check for any changes that might have occurred in the process since this book was written.
{ewc mvimg, mvimage,!intjump.bmp}

{ewc mvimg, mvimage,!illust.bmp}
Figure 4-1. The first page of the InterNIC domain registration form.

Note We've seen it take from as little as one day to as long as three weeks for the process of registering your domain name to be completed. Certain kinds of domains take longer to get because of the qualification process. Normally, registering within the *com* domain is fast.

It is important that you consult the Windows NT Hardware Compatibility List (HCL) before you choose network hardware and components for your Internet site. Hardware that is on the HCL has been tested to work with Windows NT Server, and, in many cases, the driver for that particular hardware will be included in Windows NT Server, making it easier to install. If the driver for your hardware is not included in Windows NT Server, then the drivers come with the hardware, and you will have to install them separately, either during the installation process or later.

After you have purchased your network hardware, be it modems, ISDN adapters, frame relay or network cards, you will have to install it in your machine. Of all the things you need to do to get your Internet site set up, installing network hardware for frame relay or leased line is likely to be the most difficult task you will encounter. Our advice to you here is to get professional help if you need it.

Because so many different brands are available and the steps for setting up this hardware vary greatly depending on the make, we are going to have to punt this topic and tell you to consult the documentation that comes with your hardware for the installation instructions. It would take us volumes to go through the steps for setting up this hardware for each and every brand. If you have a good ISP, you should be able to rely on their help in both choosing the network hardware and setting it up. If your ISP cannot provide these services, we strongly recommend that you hire a network consultant.

Whether you will be doing a fresh installation of Windows NT Server or configuring the settings of a previously installed server, in order to set up your Internet site you'll need the following information. You must consider such issues as which protocols to install, which file system to use, what role your server will play within Windows NT Server terminology, and which additional services will be applicable to your Internet site. Windows NT Server setup is comparatively straightforward, but taking the time to prepare for some of the questions that will come up in the installation will be of benefit to you.

This section contains the following topics:

To set up a server for the Internet, you must use the TCP/IP protocols. Windows NT Server also supports several other protocols for networking. Two of the most common protocols are built into the operating system. They are IPX and NetBEUI. If you do not have a LAN and are setting up only a single machine to be your Internet server, you should not load any protocol other than TCP/IP because each additional protocol you load uses memory and processing resources. If you do have a LAN, you can use TCP/IP for your entire network unless you are concerned with the potential security risk of having all your machines accessible via the Internet. If you don't want to use tcp/ip on all your computers, you should load whichever protocol is necessary to communicate with your other computers. We recommend that you use IPX as your first choice after tcp/ip.

Windows NT Server supports two file system types on hard disks, NTFS (Windows NT File System) and FAT (File Allocation Table). A *file system* describes the way a hard disk is formatted and the way it interacts with the operating system. You need to choose which file system you will use on your server.

The FAT system is an old file system with limited functionality. The only advantage to using FAT is that if you *dual boot* your server, which gives you the option of loading either MS-DOS or Windows NT Server, you will be able to see the hard disk when you are in MS-DOS. FAT can't support file-based security, does not have the fault tolerance capabilities that NTFS has, and doesn't handle block allocation as well as NTFS does on large drives.

Note Block allocation refers to the way in which a computer divides a hard disk into discrete chunks where the data is placed. A file can span more than one block, but a single block can't have more than one file in it. Smaller blocks mean less wasted space for very small files.

The main advantage to NTFS over FAT is that it provides you with significant security enhancements. If you use NTFS, security can be set on files, directories, and drives by user and by group, allowing significant security flexibility. Also, an NTFS drive is secure if someone moves the drive to another machine, reinstalls Windows NT Server, or boots into another operating system, because without the proper user accounts, the system will not let anyone in. Because of its security capabilities, we strongly recommend that you use NTFS as the file system for your Internet server.

Windows NT Server can function and be installed in one of three possible roles: primary domain controller, backup domain controller, and stand-alone server. In addition, a stand-alone server can be either a member of a domain or a member of a workgroup. A *domain controller* is responsible for logon security for all the machines within a given domain. The *primary domain controller* has the master copy of the user database. A *backup domain controller* has a copy of the user database and helps authenticate users for the domain when the primary one is busy. You can assign only one primary domain controller per domain, but you can have any number of backup domain controllers. A *stand-alone server* doesn't have a copy of the domain's user database and doesn't handle authentication for the domain, which gives the server a slight performance advantage over a domain controller.

If your Internet server is your only computer, you should make it the primary domain controller. If you plan to connect your Internet server to a network that already has a Windows NT domain, you can install Windows NT Server either as a backup domain controller or as a stand-alone server in the domain. You could also choose to make the Internet server a stand-alone server in a workgroup, but being outside a domain makes some of the security options that we discuss in the next chapter unavailable.

Each additional service you install on your Internet server will consume system memory and processing time. As the volume of traffic on your Internet site increases, you'll find that you need every bit of memory and processing power you have. Therefore, as you consider the following additional services, make sure that you install additional services only if you absolutely need them.

DNS

To refresh your memory, DNS is the service that translates your machine's IP address into a domain address. You can set up your domain name several ways. You can either maintain your own DNS server or have your ISP maintain DNS for you.

If you decide to maintain your own DNS server, you need to install the DNS server that comes with Windows NT Server. This would give you the advantage of being able to make immediate changes to your domain as opposed to having to notify your ISP and wait for the ISP to make the changes. One warning before you proceed: maintaining a DNS server on the Internet is complicated. DNS has a specific file format that it uses for its database. This format can be very difficult to understand, and it is beyond the scope of this book to show you how to create DNS databases. Because of its complexity, we recommend that you use your ISP's DNS server. If you would like more information about DNS, refer to our recommended reading list in Appendix C.

So, if you don't want to spend the time to maintain your own DNS, you can get DNS service from your ISP in one of two ways. One option is to have a third-level domain name off your ISP's domain. For example, if you wanted to use the domain name *kahn* and your ISP was GUIware, your domain address would be *www.kahn.guiware.com*. Your ISP probably charges a small fee for the service. Another option is to have your own second-level domain name hosted on your ISP's DNS server. In this case, your domain address would be *www.kahn.com*. Besides having a second-level domain name, this has the advantage of passing the administration of your DNS server to your ISP. Your ISP probably charges a higher fee for this, but it may well be worth it if you don't know how to manage a DNS system.

Dynamic Host Configuration Protocol (DHCP)

DHCP is a service that you will need to consider only if you are going to configure TCP/IP on a large number of workstations. DHCP dynamically assigns TCP/IP settings, such as IP address, subnet mask, and default router to workstations as they are needed. If you have a network connected to the Internet, you would configure a *scope,* which consists of a range of TCP/IP addresses and their respective settings. When one of the computers connects to the network, it asks the DHCP server for an available TCP/IP setting. DHCP can be useful with the management of TCP/IP addresses for a large network, but it is not necessary. For a small number of workstations, setting up DHCP is probably not worth the effort. If you have only one computer that will communicate with the Internet, you should not install the DHCP server.

Windows Internet Naming Service (WINS)

WINS is another name-resolution service like DNS, which has a distinct advantage if you are using DHCP. In a Microsoft networking environment, WINS doesn't require the administrator to enter TCP/IP information for each workstation in the database. As a workstation boots up on the network, it will register itself with the WINS server. If you are using DHCP to configure your TCP/IP properties, it's possible that the IP address for a computer might change from day to day. If this happens, a DNS server would have to be manually updated to track the new IP address. WINS, on the other hand, gets the new IP address from the workstation when it acquires a new IP address. If you have only a single computer or a small network of computers, you might not want to install WINS. If you have a large network, you should use WINS. You should definitely use WINS if you are using DHCP.

Routing

If you have a LAN that you want to connect to the Internet, you need to have a router to direct traffic from the Internet to the individual computers on your network. You don't need a router for a single computer connecting to the Internet.

If you need to use routing, you can do it two ways. You can use the router built into Windows NT Server, or you can set up a dedicated router. The router in Windows NT Server is an economical routing solution for low-traffic situations. If you are expecting a high volume of traffic, you probably want to buy a dedicated router.

FTP Server

If you decide to have an FTP server, you can take two approaches. You can install the FTP server that is included in Windows NT Server, or you can use the FTP server that comes with the Internet Information Server. If you plan to use any of the other pieces of IIS, such as WWW and Gopher, we recommend you use the FTP service in IIS. The FTP service included in IIS provides more functionality, better security, enhanced logging, and superior performance to the FTP service in Windows NT Server's TCP/IP. If you plan to use the FTP server that is part of IIS, it will be automatically installed along with IIS. If you decide to use the FTP server that is included in Windows NT Server, you will select it from the list of additional network services you see during installation.

Note If you are installing the IIS FTP server, don't select the FTP server that is part of Windows NT Server.

Let’s make sure you’re ready to proceed. Here is a checklist that includes all the steps you should already have taken to be ready to set up your server.

® Your computer hardware is ready, and the communications hardware is installed in it.

® You have chosen the services you want to have on your site.

® You have contracted with an ISP and have been given your IP address or address range, subnet mask, default gateway, DNS server names and their IP addresses, and any other information you will need to set up your site.

® If the ISP didn’t contact the telephone company on your behalf, then you have contacted them yourself and made sure that you have the actual communication line set up and the information you need to configure the connection.

If you are going to do a fresh installation of Windows NT Server, you can configure a good percentage of the settings for your Internet site during that process. In the following section, we'll walk you through setup for Windows NT Server. We don't intend this to be a replacement for the Windows NT Server documentation — merely a supplement that contains advice specifically for setting up your Internet server. For complete details regarding the Windows NT Server installation process, refer to the documentation that came with the operating system.

We have divided Windows NT Server Setup into smaller sections to make it more manageable and easier to talk about. The initial setup processes occur in text mode, and these include all the procedures necessary to set up your disks. When the text mode part of Setup is complete, the server software will have been installed to a point where it reboots the computer and then switches to graphics mode. Within graphics mode, three primary areas of Setup need to be completed: gathering information about your computer, installing Windows NT Networking, and finishing Setup. Installing Windows NT networking is the most critical section of the installation for someone setting up an Internet server. This is also the most complex part of the installation process.

Before you set up Windows NT Server, you might want to read through this section and take a look at the figures of the Setup screens before you actually start the installation. Anything that is not covered here you will be able to find in the documentation for Windows NT Server. If any of the steps produce results different from what we tell you here, consult the documentation for Windows NT Server installation.

This section contains the following topics:

1. Boot Windows NT from the boot disks. At the Welcome To Setup screen, press Enter.
   {ewc mvimg, mvimage,!illust.bmp}
2. Follow the prompts for detection of SCSI devices such as hard disks and CD-ROM. If Setup is unable to detect your SCSI devices, you should make sure that your SCSI adapter is configured properly and is on the HCL for Windows NT Server. Press Enter to continue.
3. Verify that the machine-specific information displayed is correct for your computer, and press Enter.
   {ewc mvimg, mvimage,!illust.bmp}
4. Follow the prompts to partition and format your hard disk. Our recommendation is that you create one partition for the entire hard disk and format it as NTFS.
5. Specify the directory where Windows NT Server will be installed. We recommend that you accept the default of WINNT.
6. Setup will check your hard disk for disk errors that might cause problems during setup. We recommend that you allow setup to do this.
7. Setup will then start copying the system onto your hard disk. Have fun watching the yellow bar get longer.
8. When the system has finished copying, it will prompt you to restart. The system automatically restarts when you press enter. After the system has restarted, you will be in graphics Setup mode.

Note The boot loader will appear for a few seconds at the beginning of the restart. No action is necessary on your part.

---

1. The first screen you see is the Software License Agreement page. You must accept the agreement in order to proceed with installation. Choose the Yes button.
2. Setup then initializes and copies some files. When the copy process is complete, you will be in the Setup wizard. Choose the Next button.
   {ewc mvimg, mvimage,!illust.bmp}
3. At the Name And Organization page, enter your name and your company name. This information is displayed in most of the administrative tools in Windows NT Server. Choose the Next button.
4. At this point, you need to determine the licensing mode you want Windows NT Server to operate in. Please refer to online help for more information. Select your licensing mode, and choose the Next button.
   {ewc mvimg, mvimage,!illust.bmp}
5. Windows NT requires a unique name for your computer. This name is used only by computers on your LAN and not on the Internet. Type a computer name, and choose the Next button.
6. You must select the role for your Internet server. For more information about roles, refer to the section on this topic on page 52 of this book and to the Windows NT Server documentation. Select the role, and choose the Next button.
   {ewc mvimg, mvimage,!illust.bmp}
7. If you selected Primary Domain Controller or Stand-Alone Server, you will be prompted for a password for your Administrator account. Every Windows NT Server machine you install has a local administrator. The password being asked for here is for that user. Type in a password and keep it safe. Choose the Next button.
   {ewc mvimg, mvimage,!illust.bmp}
8. An emergency repair disk can be used in the event your hard disk is corrupted. You don’t need to create an emergency repair disk during setup because you can do this at any time using RDISK as noted in the dialog box. Although your screen recommends Yes, we recommend that you create an ERD at a later time. Select No and choose the Next button.
9. At this point, Windows NT Server Setup prompts you for additional components to install. We recommend that you accept the default components. You might want to install additional ones later. Refer to your documentation for further information. Choose the Next button.

Congratulations! You have just completed the easy part of the installation process. Welcome to the Network Installation informational screen. You now enter the part of Setup where you will configure settings for your networking components.

1. Choose the Next button to continue.
   {ewc mvimg, mvimage,!illust.bmp}
2. Setup needs to know how this computer will communicate with both the Internet and any LAN you might have. If you plan to have your internet server connect directly to the Internet using anything other than a dial-up connection, you should select the Wired To The Network option. Additionally, if you have a LAN, you should select Wired To The Network. If you plan to use dial-up networking, either to connect your site to the Internet or to have people connect to you, you should select Remote Access To The Network. These options are not mutually exclusive, so you can select both. Choose the Next button.
   {ewc mvimg, mvimage,!illust.bmp}
3. Now Setup asks you whether you want to install Microsoft Internet Information Server. This is the heart of your Internet server, so we recommend you select this option. (No kidding!)
   {ewc mvimg, mvimage,!illust.bmp}

Note If you checked the Wired To The Network option, continue with Step 4. If not, proceed to Step 7.

---

4. Setup needs to determine what kind of network adapters you have in your computer. In many cases, setup can automatically detect these devices. But if you are using a device that is not on the hardware compatibility list or is older and therefore doesn't support autodetection, you might need to select it manually. You should choose the Start Search button to see whether Setup can find your adapters.
5. If Setup finds an adapter, it will list it. If you have more than one adapter, you should choose the Find Next button until Setup has found all of your adapters.
6. To manually select an adapter, choose the Select From List button. If the adapter is shown in the list, choose it; otherwise, you need a disk from the adapter's manufacturer with the Windows NT drivers on it. If you have the disk, choose the Have Disk button and follow the prompts.
7. Now you get to choose your Network Protocols. Assuming that you installed the IIS, TCP/IP will automatically be selected in this dialog. Any other protocols should be selected only if you need them. Choose the Next button.
   {ewc mvimg, mvimage,!illust.bmp}
8. Setup now needs to know what additional network services you want to install on your Internet server. It shows you a list of the services that are currently included.
   {ewc mvimg, mvimage,!illust.bmp}
9. To install additional services, choose the Select From List button and then select the specific service you want to add. If you want to install more than one additional service, you must repeat this process. When you finish, choose the Next button.
10. You see another informational screen that tells you that you are ready to install networking components. Choose the Next button.

Depending on the type of network cards you have in your computer, you might see additional dialog boxes that ask you to select specific options for those cards.

1. Follow the prompts, and refer to your network adapter's documentation if you require assistance. When you are done, choose the OK button.
2. At this point, Setup will ask you how you want to configure your TCP/IP settings. The first question it asks is whether you have a DHCP server on your network. If this is your only machine, or if you know you don't have a DHCP server, click No. Otherwise, if you have a DHCP server and it is configured to give out valid Internet IP addresses, select Yes.

Note If you checked Dial-in Networking, proceed with Step 3. If not, go to Step 8.

---

3. At this point, Setup attempts to configure your dial-in and dial-out capabilities. If Setup is unable to find compatible communications hardware, you'll be asked to invoke the modem installer. If you have a modem, you should click Yes and follow the prompts from the Install New Modem wizard.
4. If Setup detects your communications device, or if you manually select it, Setup then prompts you to select the RAS-capable devices you want to use. Once you have done this, click the OK button.
   {ewc mvimg, mvimage,!illust.bmp}
5. In the Remote Access Setup dialog box, you should select the Configure button for each communications device that you have.
   {ewc mvimg, mvimage,!illust.bmp}
6. A communications device can be configured one of three ways. *Dial Out Only* means that the specified device will be used to connect the server to some other network such as the Internet. *Receive Calls Only* means that the specified device will be used to allow other people to connect to your server. *Dial Out And Receive Calls* allows the specified device to be used for both dialing in and dialing out. Select the appropriate option for each communications device that you have.
   {ewc mvimg, mvimage,!illust.bmp}
7. After configuring your devices, click the Network button in the Remote Access Setup dialog box. In the Network Configuration dialog box, make sure that at least the TCP/IP protocol is selected. Other protocols should be selected only as needed. When you get back to the Remote Access Setup dialog box, click the Continue button. Setup now installs the networking components.
   {ewc mvimg, mvimage,!illust.bmp}
8. If you don't have DHCP, Setup prompts you for TCP/IP configuration information. You need to enter the IP address for your Internet server along with its subnet mask and default gateway. If you have more than one IP address you want to assign to this machine, you should click the Advanced button to enter them.
   {ewc mvimg, mvimage,!illust.bmp}
9. In the Advanced IP Addressing dialog box, type in the IP address and subnet mask and click the Add button for IP addresses for each IP address you want to include. When you finish, click OK.
10. Select the DNS tab of the TCP/IP properties dialog box. Enter the host name you want to use for your server on the Internet. This name doesn't have to be the same as your machine name, but it will default to the machine name. Then enter the Internet domain name for this server. Click the Add button for DNS Service Search Order, and enter the IP address for your DNS server. Either your ISP will have provided you with this address or it will be your own IP address if you set up your own DNS server.
    {ewc mvimg, mvimage,!illust.bmp}
11. If you are using your Internet server as a router, select the Routing tab in the TCP/IP Properties dialog box and check the Enable IP Forwarding box.
12. If you have WINS installed on your network, configure the WINS Address tab of TCP/IP Properties as well. You don't need this for your Internet server so we will not go into detail describing this service.
13. When you have finished configuring the TCP/IP properties, click the OK button. If you didn't configure WINS, Setup will ask you to verify that this is OK. Click the Yes button. Some processing will occur.
14. Setup displays the network bindings about to occur. A binding connects a protocol to a service and to an adapter. We highly recommend that you accept the default bindings as Setup configures them. Click the Next button.
    {ewc mvimg, mvimage,!illust.bmp}
15. Setup is now ready to start your network services. Click the Next button.

16. Twiddle your thumbs for a few moments while Setup does some processing.
17. Depending on the role you selected for your server, you see a dialog box that either creates a new domain or joins a backup domain controller into an already existing domain. Or, if you chose to make yours a stand-alone server, you choose from options to join a domain or a workgroup. You should either create or join a Windows NT Server domain. Click the Next button.

You are almost through the setup process.

1. Click the Finish button to continue the process. Watch the system do some more processing.
   {ewc mvimg, mvimage,!illust.bmp}
2. Setup will now walk you through the installation of IIS. Select the services you want to provide on the Internet from the list shown. By default, all services are selected. You need Internet Service Manager and Help & Sample Files. ODBC Drivers & Administration is necessary if you plan to offer any database connectivity to your Internet server. You can choose to remove the FTP, Gopher, and WWW services if you don't need them. Then click the OK button. Confirm the creation of the directory where IIS will be installed by clicking Yes.
   {ewc mvimg, mvimage,!illust.bmp}
3. The IIS needs a location for the content of the different services that you plan to offer. Lists of contents like these are known as *publishing directories*. Setup suggests default directories, or you can choose your own.
   {ewc mvimg, mvimage,!illust.bmp}
4. Click the OK button once you have specified the directories. If these directories don't yet exist, confirm their creation by clicking Yes. Watch the system do some copying. Stretch your legs. Get a cup of coffee.
5. If you selected ODBC, Setup now prompts you to install drivers for ODBC. Select the appropriate driver for your database. For example, if you have Microsoft SQL Server, select SQL Server from the list. Click the OK button. See even more copying!
6. Select your time zone from the list. Select the Date/Time tab to verify the correct time and date. Click the Close button.
   {ewc mvimg, mvimage,!illust.bmp}
7. Setup automatically identifies the kind of video card that you have and now prompts you to configure the settings you want to use. Click OK in the Detected Display dialog box, and choose the settings you want. Follow the prompts to set your video settings. Click OK when you're done. Now see some more delightful processing. Isn't it fun watching those bars go by? This is a good time to take up a new hobby, say knitting.
   {ewc mvimg, mvimage,!illust.bmp}
8. Windows NT Server is now installed. Choose the Restart Computer button to reboot, and take a well-deserved break. When you get back, you'll be ready to configure your system.

If you have Windows NT Server installed but not set up for communication with the Internet, you need to reconfigure some of your current settings. You should be using Windows NT Server version 4.0 or later for your Internet site. If you have an older version, the first thing you should do is upgrade to 4.0. If you need to upgrade, refer to the documentation that comes with the upgrade. The following describes the process for getting a previously installed Windows NT 4.0 Server ready to configure for the Internet.

Checking Your Network Configuration

1. The first thing you should do is determine whether TCP/IP is already installed and properly configured. You need to be logged in as Administrator or as a user with administrative privileges to configure these areas.
2. Launch the Network control panel found inside the Control Panel.
3. Select the Protocols tab.

Note If you find that TCP/IP has already been installed, continue with Step 4. If you need to install TCP/IP, proceed to Step 6.

---

4. If TCP/IP is in the list, that protocol is already installed. You should check its properties by clicking the Properties button.
5. Compare your TCP/IP properties with the setting your ISP gave you. Make any changes as necessary.
6. If TCP/IP is not in the list, click the Add button and select TCP/IP Protocol from the list. Click the OK button.
7. The system asks whether DHCP is used. Unless you actually have set up a DHCP server on your network, click the No button.
8. Next you need to insert the installation CD for Windows NT Server. TCP/IP will then be listed as one of the protocols. Click the Close button.
9. Processing happens, and then you will see the Microsoft TCP/IP Properties dialog box. At this point, you must enter your IP address, subnet mask, and default gateway.
10. If you have DNS server information, you should click on the DNS tab and choose the Add button under DNS Service Search Order. Enter the IP address of each DNS server, and click the Add button. If you have more than one adapter in your server, you must enter this information for each adapter. Make sure that you enter the right information from the right adapter. Click the OK button when you're done.
    {ewc mvimg, mvimage,!illust.bmp}
11. You might get a message that tells you the primary WINS address is empty. Click the Yes button to continue.
12. Setup now requests that you reboot your computer for the changes to take effect. Do it.

Install the Internet Information Server (IIS)

You need to install the Internet Information Server Version 2.0, which is Microsoft's Web, Gopher server, and FTP service.

1. To install IIS, insert the Windows NT Server CD.
2. Open a command prompt, and change to the CD drive.
3. Change directories to the inetsrv subdirectory.
4. Execute the INETSPT.EXE program by typing *Inetstp*. When you see the IIS informational screen, click the OK button.
5. Make sure that all the services you want are installed. Refer to the Fresh Install section under Finishing Setup for information about the services and the steps to complete the installation of IIS.

Check for Your Communications Devices

You must make sure Windows NT Server can communicate with the Internet.

1. Launch the Network control panel from inside the Control Panel.
2. If you plan to use a modem or ISDN, select the Services tab and add Remote Access Service if it's not already installed. To configure Remote Access Service, follow the steps in the Install Networking Components section on page 62.

3. If you plan to use frame relay or a leased line with an adapter in your Windows NT Server, select the Adapters tab and click the Add button to install the driver for your communications card. Refer to the documentation that came with your adapter for more information on installing the drivers.

Note For ISDN, you may also have to go to the Adapters tab and add the ISDN card. See your ISDN documentation for more information.

Once you complete a fresh installation or modify a previously installed server, you need to do some additional configuration in order to complete the setup and establish your Internet link. In this section, we will walk you through the steps to configure IIS and DNS and to connect to the Internet. When you have completed this, the only areas left for consideration will be security configuration and directory layouts.

This section contains the following topics:

Setup installs IIS with default settings that might not be appropriate for your Internet site. Before going live on the Internet, you should go through the Internet Service Manager to check the settings and make any changes that you need. Internet Service Manager can be found in the Programs/Microsoft Internet Server group.

If you have enabled all the services on IIS, specifically WWW, FTP, and Gopher, you need to configure settings for each of these services. We'll go through the complete process of configuring Web for IIS, meaning that we discuss in this section each option for each tab. When you configure FTP and Gopher, you'll notice that many of the options are the same. We discuss only the settings that are unique to FTP and Gopher; you can refer back to the WWW section for information regarding the rest of the settings. Unless we tell you otherwise, the settings on each of the service's tabs are specific to that service even if they have the same name. This means that you must configure the setting separately for each service.
{ewc mvimg, mvimage,!illust.bmp}

This section contains the following topics:

If you installed WWW, double-click on the line for your WWW service in the Internet Service Manager to get to the settings. This dialog box has four tabs, which are for Services, Directories, Logging, and Advanced settings. You should step through each tab to ensure that you are happy with the default setting and change anything that you want to work differently.

On the Services tab, you can change the following settings:

® Connection Timeout refers to the amount of time a connection can remain idle before the server disconnects it. The default of 900 seconds in most cases is fine. You might lower this number if your server has heavy traffic.

® Maximum Connections is the total number of simultaneous connections that the server can handle. The proper setting for this depends greatly on the speed of your hardware and the amount of memory you have. The default setting is 1000 simultaneous connections. If you find users complaining about the speed of the connection at certain times, you might lower this number.

® Anonymous Logon has two items: username and password. This is the user ID and password that IIS uses to access the files on your Web site for your Web users. You should not change the default of *IUSR_machinename*.

® Password Authentication has three options: Allow Anonymous, Basic (Clear Text), and Windows NT Challenge/Response. These are the methods by which users can connect and identify themselves to the IIS server. Allow Anonymous means that users don't have to identify themselves to the server. Basic means that users will be asked for their user IDs and passwords and that information will be transmitted to the IIS. Windows NT Challenge/Response allows Windows 95 or Windows NT Workstation clients running Microsoft Internet Explorer 2.0 or later to identify themselves to the server automatically. For more information regarding security, refer to Chapter 5 of this book.

® The Comment box is used only for identification purposes via the Internet Service Manager. You can put any identifying information you want in here.
{ewc mvimg, mvimage,!illust.bmp}

Under the Directories tab, you can change the following settings:

® The directory list names the directories and virtual directories that the IIS can access. The default directories are WWWROOT and SCRIPTS subdirectories. Chapter 6 covers options for configuring your directory structure for your Internet site.

® Enable Default Document means that if a user connects to your Web site without referencing a specific WWW file, the specified document will be sent to the user. By default, the Enable Default Document check box is checked, and the Default Document Name is DEFAULT.HTM. (We get a bonus for each time we use the word *default*!)

® If enabled, the Directory Browsing Allowed option lets the user see all documents on your Web server in a file list and select from it. The default setting for this option is disabled, and we recommend that you leave it that way to prevent unintended access to files.
{ewc mvimg, mvimage,!illust.bmp}

Next select the Logging tab. Logging is the process the server uses to track use of your Internet site over time.

® The first option is Enable Logging, which is turned on by default. If you don't want to track your server usage or potential security problems, you can disable logging — it does use up both disk space and processing power. Most site builders, however, will probably want this feature.

® If you have enabled logging, you must choose between logging to a file or logging to a database. If you want to log to a database, refer to the documentation. We recommend logging to a file because most of the IIS log analysis tools work only against files.

® The Automatically Open New Log option allows the IIS to create a new log file based on the criteria you select; otherwise, one log file will be created. By default, this option is enabled to create daily log files. We recommend that you accept the default.

® Log File Directory refers to the location in which log files should be placed.
{ewc mvimg, mvimage,!illust.bmp}

Typically, you don't need to change anything on the Advanced tab unless you have specific IP addresses to which you want to grant or deny access to your site.

® You can choose to grant or deny access to all computers. The default is to grant access, and we recommend

that you accept this setting. For more detailed information, refer to the online help and documentation.

® Limit Network Use By All Internet Services On This Computer applies to FTP and Gopher as well as the WWW service. It enables you to throttle the bandwidth that the Internet services use. We recommend you don't enable this option.
{ewc mvimg, mvimage,!illust.bmp}

When you finish reviewing and making any necessary changes to your WWW settings, click the OK button.

If you chose to have an FTP site on IIS, double-click on the entry that corresponds to FTP in the IIS Manager. You see tabs for Service, Messages, Directories, Logging, and Advanced settings. Some of these tabs have settings similar to those in the WWW settings. We're walking you through only the exceptions, here.

- On the Service tab, you find two items that differ from the WWW settings. Allow Only Anonymous Connections prevents users from identifying themselves to the system; they can connect only as anonymous users. If you want to provide special access to certain users, you might want to change that default setting from enabled. The Current Sessions button displays the currently logged-on FTP users on your IIS. You can disconnect individual users or all users from the FTP service.
  {ewc mvimg, mvimage,!illust.bmp}
- The Messages tab is unique to the FTP service and allows you to specify messages to be displayed to your users at logon, logoff, and when your server is busy. Type in any messages that you want. If you leave these blank, no message will be displayed.
  {ewc mvimg, mvimage,!illust.bmp}
- Under the Directories tab, you find Directory Listing Style. The two options are UNIX and MS-DOS. Each option refers to the way the contents of a directory (names, dates, and size of files, for example) are displayed to the user. The default is MS-DOS; this setting is a personal preference so we make no recommendations.
  {ewc mvimg, mvimage,!illust.bmp}

When you finish reviewing and changing the settings, click the OK button to return to the Internet Service Manager.

When you have an Internet site, two of your main concerns are that your system perform adequately and that it be available 24 hours a day. By configuring certain system and network properties, you can improve the performance and availability of your server. If you want a truly thorough discussion of how to optimize your Internet server, you should refer to the Windows NT Resource Kit. But we'll give you *some* guidance here.

This section contains the following topics:

System settings control general behavior of Windows NT Server. By changing some of the system properties settings, you can improve the performance of certain applications and provide your server with default actions to take when certain conditions are met.

1. Launch the Systems control panel from the Control Panel.
2. Select the Performance tab, and change the Application Performance Boost setting to None. When Application Performance Boost is set to Maximum, the system gives the priority for system resources to the foreground applications running on your server. Since your Internet-related services are considered background applications, changing the setting to None means that the foreground applications are given no priority over background applications.
3. Click the Change button for Virtual Memory, and verify that Initial Size for Page File Size For Selected Drive is set at the minimum recommended page file size.
   {ewc mvimg, mvimage,!illust.bmp}
   {ewc mvimg, mvimage,!illust.bmp}
4. Select the Startup/Shutdown tab in the System Properties dialog box.
5. Reduce the number of seconds in the Show List For option to 10 seconds. This decreases the amount of time the system waits while rebooting before loading Windows NT Server.
6. Choose the Automatically Reboot option to enable Windows NT Server to reset itself in the event of an operating system failure.
7. Click the Close button to close the System Properties dialog box.
   {ewc mvimg, mvimage,!illust.bmp}

If you installed more than one protocol, you can get a slight performance boost for your Internet traffic by setting the TCP/IP protocol as your top-level protocol.

1. Launch the Network control panel from the Control Panel.
2. Select the Bindings tab, and make sure that the Show Bindings For option has All Services selected.
3. Click the plus signs to open the protocol order list for each service. If TCP/IP is not the first protocol listed, highlight it and click the Move Up button until it is.
4. Repeat this process for each service listed.
   {ewc mvimg, mvimage,!illust.bmp}

Without a working connection, everything you have done up until now is pointless. Depending on the type of connection you are using, you might need to do some additional configuration to get the connection up and running. Getting your connection to work reliably can be one of the most challenging steps you make in this journey.

This section contains the following topics:

If you are connecting to the Internet via modem or ISDN, you need to give Windows NT Server the appropriate phone number and logon credentials to use.

1. Double-click the Dial-Up Networking icon found on your desktop.
2. Click the New Entry button to create the connection to your ISP, and enter a name for this connection. Click the Next button.
   {ewc mvimg, mvimage,!illust.bmp}
3. Click the I Am Calling The Internet check box. You might need to select the other two options on this screen depending on your ISP. If you are connecting via PPP, you shouldn't have to select any additional settings on this screen. Send My Plain Text Password If That's The Only Way To Connect refers to the ISP's ability to support encryption of your password. The ISP will tell you if you need to select this. You use the third option when a SLIP connection is required.

   Note Point-to-point Protocol and SLIP are communication protocols used for asynchronous communication to a network. SLIP is a much older and less advanced protocol that very few ISPs still support. You should make every effort to use PPP when connecting to the Internet.

   ---

4. Click the Next button, which takes you to the Modem Or Adapter selection screen. Select the device you plan to use to connect to the Internet.
   {ewc mvimg, mvimage,!illust.bmp}
5. Click the Next button, which brings you to the Phone Number page. Enter the phone number your ISP gave you for your account. If you have more than one number, click the Alternates button and add them to the list.
6. Click the Next button and then the Finish button, and your entry is saved.
7. In the Dial-up Networking dialog box, select the dial button to initiate the connection. You probably want to click the More button and choose User Preferences. In the User Preferences dialog box, you want to set the number of redial attempts to a high number, such as 1000, the seconds between redial to 3, and the idle seconds before hang-up to zero. This is because a phone connection is not as stable as a dedicated connection and could drop the line, thereby cutting you off. With these settings operating, your server will automatically redial if it loses connection.
   {ewc mvimg, mvimage,!illust.bmp}
8. Repeat this process for logon preferences.

If you installed a frame relay or a leased adapter in your server, you already configured the settings for these connections during the installation process or when you modified your server. If you plan to use a dedicated router to connect your network to the Internet, you will need to refer to the documentation that came with your router and talk to your telephone company to configure your connection.

A couple of common problems could prevent your connection from working properly. First make sure that the phone company has activated the link, sometimes referred to as “turning up the link.” Then ensure that you have the correct parameters set on your system for the type of equipment the phone company uses. For example, the link-management protocol must be configured properly and the routing structure must be configured to support the encapsulation that is required by your ISP. Many different settings must be correct for frame relay and leased lines to work properly. You will have to depend on help from your phone company, your ISP, and the manufacturer of your communication equipment to configure your specific setup.

Once your server is operational and connected to the Internet, you should test the connectivity prior to announcing your server. One way to test your server is to use it as a client to surf the Web. Using Internet Explorer, you should attempt to connect to other Web sites, FTP sites, and Gopher sites, and you should try to "ping" other Internet servers from your Internet server. Ping is a utility that attempts to make a connection with another computer by sending a packet of information and keeps track of the amount of time it takes to make the connection. If the ping is successful, you will receive a message to that effect on your command-line prompt. In order to ping, you should go to a command prompt and type the command PING *hostname*; for example, PING *www.guiware.com*.

The other method for testing connectivity is to attempt to connect to your Internet server from the outside. Use a friend's computer, or find some other way to get outside access to the Internet, and then try to ping your server and get access to your Internet services.

In Chapter 3, we recommended that everyone use backup and that you purchase a SCSI tape drive as your backup device. Once again, to avoid the possibility of massive frustration, we advise that you acquire a backup device that is on the HCL. Follow the directions that come with this device in order to hook it up to your machine. Then install the drivers within Windows NT Server.

We recommended previously that only those who are really concerned about the possibility of downtime use fault tolerance, specifically disk mirroring. Disk mirroring maintains a redundant copy of your data on a separate disk to protect you from the possibility of losing your data due to disk failure. You must have two disks to set up disk mirroring, and the disk that is to act as the mirror must have at least the same capacity as the original disk.

Installing Backup

1. Open Tape Devices in Control Panel.
2. Choose the Detect button. The system will attempt to detect your tape device. If successful, you will see the name of your device in the dialog box.
3. If for some reason the system is unable to auto-detect your device, you can install it manually. Select the Drivers tab, choose Add, and select your tape device from the list. Choose Have Disk if you have a vendor-supplied driver disk.
   {ewc mvimg, mvimage,!illust.bmp}

Running Backup

1. Launch Backup from the Administrative Tools program group. You will see all the drives that you are connected to.
2. If you want to back up an entire drive, select the check box that corresponds to that drive.
3. If you want to back up only specific directories or files, double-click on the drive where the directories or files are found and use the disk tree to select the resources that you want to back up.
4. When you have finished selecting the resources that you want to back up, choose the Backup button.
5. Follow the prompts to name the backup set, and choose the OK button to start the backup.
   {ewc mvimg, mvimage,!illust.bmp}

Note The built-in backup software for NT can reliably back up only the local machine and not multiple machines on your network. If you want a network backup solution, you should look at one of the several third-party backup solutions for Windows NT Server.

Creating a Disk Mirror

1. To establish a mirror set, launch Disk Administrator from the Administrative Tools program group.
2. Highlight the partition you wish to mirror.
3. Hold down the Control (Ctrl) key, and highlight free, unpartitioned space on the second disk.
4. Select Establish Mirror from the Fault Tolerance menu.
   {ewc mvimg, mvimage,!illust.bmp}

In the Event of a Disk Failure

To make use of the mirrored partition, you must first break the mirror set that you created.

1. Launch the Disk Administrator from the Administrative Tools program group.
2. Highlight the mirrored partition, and choose Break Mirror from the Fault Tolerance menu.
3. Exit Disk Administrator, and reboot Windows NT Server.

Mirroring Your Boot Partition

Your boot partition is a special case because, in order to activate a mirrored partition, you must be able to run Disk Administrator, which would require Windows NT Server to boot. There is a process to create a boot floppy

to handle this situation. Refer to the Concepts and Planning Guide in the Windows NT Server documentation for more details.

Installing IIS and configuring Windows NT Server for the Internet gives you a limited number of Internet services that you can have on your site. If you want to expand the functionality of your Internet presence with the addition of databases, e-mail, WAIS, or other Internet services that are not included in IIS, you need to install additional software for those services.

Microsoft SQL Server

If you want any kind of forms, surveys, transactions, or dynamic data shown to your WWW users as part of your Internet site, you need to have a database to act as a repository for the information. The best choice for your database needs would be Microsoft SQL Server. Since IIS is designed to interface with SQL Server, it is the most reliable database that you can use for your Internet needs. IIS is able to access data in a Microsoft SQL Server database without requiring that you write code. It is also fairly easy to install if you follow the documentation that comes with it. We highly recommend the use of SQL Server to extend the functionality of your Internet site.

E-mail

In addition to the other services you have installed on your Internet server, you might also want e-mail so that you can communicate with users and customers. Your users could want to send you e-mail regarding any problems that they have with your Internet site, or you might use e-mail as a way for users to order products or ask you questions. As your site becomes more advanced, or if you become an IPP, you might want to rent e-mail boxes to other users or provide feedback accounts for your WWW customers.

Several e-mail systems are available for Windows NT Server. Shop around and ask some questions about different features and the way that they operate. The two e-mail systems that Microsoft has available are Microsoft Exchange Server, which is the most advanced e-mail system, and MAILSRV.

In addition to standard e-mail functions, Microsoft Exchange Server provides a rich set of features, including public folders, workgroup productivity, and links to other applications, such as Microsoft SQL Server. If you are looking for a high-end mail system that can handle a lot of traffic and provide more than simple mail services, you might want this system.

MAILSRV, which is part of the Windows NT Resource Kit, is a very basic mail system that lets you send and receive mail only. It uses POP3 and SMTP as its communication protocols and requires the use of a POP3-compliant client such as the Exchange Inbox that is found in the Windows family of operating systems. In order to send and receive mail on your Windows NT Server, you should use the Exchange client with the Internet mail service that comes with Windows NT Server. MAILSRV is very easy to install, and almost no maintenance is required.

We won’t go through the installation process for setting up e-mail because the installation routine depends on the e-mail server you use. Refer to the documentation that comes with the e-mail system that you choose.

Windows NT Resource Kit

The Windows NT Resource Kit is a collection of books and software that provides detailed technical information and utilities to support and enhance Windows NT Server. Besides MAILSRV, the Resource Kit also includes a Telnet service; a WAIS (Z39.50) search engine and the WAISTOOL used to create the index for searching; and a new tool called dbWeb that allows you to quickly publish SQL Server databases on the WWW. This is just a smattering of what you’ll find in the Resource Kit. We highly recommend that you purchase Windows NT Server Resource Kit version 4.0 and any updates; it’s the most comprehensive body of work available about Windows NT Server.

Now that the construction of your Internet server is complete, you need to address the security of your Internet site. The next chapter walks you through the various security issues and gives you suggestions about how to address them. Security is one of the most important aspects that you need to consider for your Internet server.

It’s time to get your Internet server up and running. You have some big jobs ahead of you. If you’ve already contacted an ISP, you know that you need to choose and register a domain name and set up your network hardware. Then you need to make sure your operating system is installed and configured properly, depending on the services that you choose for your Internet site. Once all this is accomplished, the big task that remains in getting your Internet server operational is setting up your link to the Internet and testing it for connectivity. Doesn’t sound too bad, does it? Well, none of this is difficult, but to be truthful, some of the steps might try your patience.

We will walk you through all processes as best we can to get your Internet server in working order. Once we’re there, we’ll show you how to test your connection and set up backup and basic fault tolerance, and we’ll discuss additional software that you can use to expand the range of services on your site. When you get through all the steps in this chapter, the skeleton of your Internet presence will be up and running and just waiting for you to add your content. It will be like having a whole bunch of empty bookshelves waiting for you to set up your library.

Note About time: We want to warn you that it might take you longer than you expect to get your server running. Setting up an Internet presence is a complex process, and even experts cannot accomplish it overnight. If you’ve already contacted an ISP, you might have learned that just the lead times for ordering your communications link could be weeks or months, and this is only one small part of setting up your server.

This chapter contains the following topics:

You build business objects to implement business services. Services can create solutions that grow with your business. A transaction is a process that occurs within services. The main purpose of a transaction is to make changes to data.

To implement transactions, you must implement a framework to manage business objects. Microsoft Transaction Server provides this framework and manages the processing of transactions for you.

Distributed Transactions

A transaction typically uses many different server components that reside on different computers. A transaction can also access multiple databases. Microsoft Transaction Server tracks components on multiple computers, and manages distributed transactions for those components automatically.

Thread Management

During a transaction, a server component can be called by multiple clients at the same time. Clients can be users or other components. When multiple threads run in the same method at the same time, concurrency problems can occur.

To avoid concurrency problems, create objects to be thread safe. Thread-safe objects require more maintenance; however, Microsoft Transaction Server provides this object maintenance for you.

If you enabled Gopher service, double-click the Gopher entry in the Internet Service Manager. The four tabs for the Gopher service are Service, Directories, Logging, and Advanced.

The Gopher service has only one unique setting, and it can be found on the Service tab. You need to type in the name and e-mail address of the Service Administrator whom users can contact when they have problems and questions.
{ewc mvimg, mvimage,!illust.bmp}

Yes, you guessed it. This chapter deals with security. Security in relation to your Internet server is a very complex topic. Not only do you need to think about guarding the information on your server from those who have direct access to it, but you also have to think about safeguarding your information from Internet users. If you are planning on doing financial transactions over the Internet or if you are sending any kind of confidential information over the Net, then you need to secure that information, too.

In this chapter, we discuss some of the possible threats to your system and how to circumvent them. Further, we show you how to implement the security features built into Windows NT Server, along with other security technologies, in order to secure your Internet site.

This section contains the following topics:

Article ID: Q162908
Creation Date: 30-JAN-1997
Revision Date: 31-JAN-1997

The information in this article applies to:

® Microsoft Access 97

Symptoms

Moderate: Requires basic macro, coding, and interoperability skills.

The data that is displayed in table Lookup fields that you created with the Lookup Wizard is ignored when you export the table to HTML, IDC, or ASP format. The bound column for the Lookup field is displayed in the Web browser, not the lookup data that you see when you open the table in Datasheet view in Microsoft Access.

Cause

If you exported to HTML format with the Save As/Export command on the File menu, you did not click to select the Save Formatted check box in the "Save Table <Table Name> In" dialog box.

If you exported to Internet Database Connector (IDC) or Active Server Pages (ASP) format, the SQL statement that Microsoft Access generated contains the actual data that is stored in the table's Lookup field. The bound column of a Lookup field's RowSource property is what is actually stored in the table, not necessarily what you see in Datasheet view of the table in Microsoft Access.

Resolution

If you want to create static HTML files, click Save As/Export on the File menu, and then click to select the Save Formatted check box in the "Save Table <`Table Name`> In" dialog box. If you click Save As HTML on the File menu to create static HTML files, the files you create from tables or queries automatically include the Save Formatted option.

If you are exporting to IDC or ASP format, create an AutoLookup query that includes the original table, as well as the table from which the Lookup field obtains its data. Use the corresponding fields from the lookup table in the query instead of the Lookup fields from the original table. Then export the query to IDC or ASP format.

For example, if you open the sample database Northwind.mdb and view the Orders table in Datasheet view, you see the complete customer name in the Customer column. However, if you export the Orders table to IDC or ASP format, only the CustomerID is exported; CustomerID is the bound column in the Customer Lookup field in the Orders table. If you create an AutoLookup query that contains both the Customers and Orders tables, you can add all the fields from the Orders table to the QBE grid, except the CustomerID field; add the CompanyName field from the Customers table instead. When you export the query, it contains the same data as the Orders table, including the full customer name.

For more information about creating AutoLookup queries, search the Help Index for "AutoLookup queries," or ask the Microsoft Access 97 Office Assistant.

More Information

u Steps to Reproduce Behavior

The following steps reproduce the behavior by exporting a static HTML file. The same behavior will occur if you export to IDC or ASP file formats.

1. Start Microsoft Access and open the sample database Northwind.mdb.
2. Open the Products table in Design view.
3. Select the SupplierID field, and then click the Lookup tab in the Field Properties section of the Design window. Information under this tab indicates that SupplierID is a Lookup field.
4. Close the Products table, and leave it selected in the Database window.
5. On the File menu, click Save As/Export.

6. In the Save As dialog box, click "To an External File or Database," and then click OK.
7. In the "Save Table 'Products' In" dialog box, select a folder on your Microsoft Internet Information Server (IIS) where you have Read permission, or save the file locally and copy the HTML file to the your IIS Web server folder after the export is complete.
8. In the "Save Table 'Products' In" dialog box, select HTML Documents (*.html; *.htm) in the Save As Type box, and then type Products.html in the File Name box. Verify that Save Formatted is not checked, and then click Export.
9. Start your Web browser program, and type the Uniform Resource Locator (URL) in the Address box to view the Products.html file. Note that the URL depends upon where your files are located on the Web Server. For example, if you saved Products.html in the wwwroot folder of your Web server, type:

   http://<servername>/products.HTML

   When the Products.html file opens, note that the SupplierID column shows the actual SupplierID code and not the Company Name.

References

For more information about saving Microsoft Access objects in HTML format, search the Help Index for "saving database objects, saving in Internet/Web formats," or ask the Microsoft Access 97 Office Assistant.

For more information about configuring Microsoft Internet Information Server permissions, please refer to the IIS Help Index.

KBCategory: kbinterop
KBSubcategory: IntpOthr
Additional reference words: 97 8.00 HTML IDC ASP export primary key

```
Set addobj = ctxObject.CreateInstance("Add.Obj")
Set dropobj = ctxObject.CreateInstance("Drop.Obj")
```

Article ID: Q161333
Creation Date: 19-DEC-1996
Revision Date: 20-DEC-1996

The information in this article applies to:

® Microsoft Access 97

Summary

Advanced: Requires expert coding, interoperability, and multiuser skills.

This article discusses some of the Microsoft Windows NT permissions issues you may encounter when you create and use Internet Database Connector (IDC) files or Active Server Page (ASP) files from a Microsoft Access database.

This article assumes that you are familiar with the Internet Database Connector, ActiveX Server, Microsoft Windows NT, and Internet Information Server (IIS).

For more information about using IDC to publish data on an intranet or the Internet, please refer to the IIS online documentation. You can also explore Microsoft's World Wide Web site for IIS at http://www.microsoft.com/iis/default.asp.

For more information about using ASP to publish data on an intranet or the Internet, please refer to the Active Server online tutorial.

Note This article does not apply to Web servers running Microsoft Windows 95 with Personal Web Server because Microsoft Windows 95 does not use the same security features that are built into Microsoft Windows NT.

More Information

If you use IDC or ASP files to publish Microsoft Access data on an intranet or the Internet, you may receive the following error message when you browse through those files with your Web browser if the permissions, usernames and passwords are not set correctly in Microsoft Windows NT:

> The Microsoft Jet database engine cannot open the file '(unknown)'.
> It is already opened exclusively by another user, or you need permission to view its data.

The following are the main reasons for the error message:

® Incorrect Username or Password

® Insufficient NTFS Directory Permissions

® Insufficient Share Permissions

® Access Database Is on a Windows 95 Computer with Insufficient Share Permissions

Each of these problems is explained below, along with the actions you must take to correct them. If you need additional assistance with permissions or user accounts, please refer to your Microsoft Windows NT or Microsoft IIS documentation.

Note The troubleshooting tips that follow apply to the anonymous logon username account set up in the IIS Internet Service Manager. IUSR_<`Server Name`> is the default account name, but it can be changed. If anonymous logons are not allowed (for example, if Basic Authentication or Windows NT Challenge/Response is in use), then the following troubleshooting tips apply to whatever user accounts may be trying to access the WWW services. For more information about setting up the WWW service in the Internet Service Manager, please refer to your IIS online documentation.

Incorrect Username or Password

You specified an incorrect username or password for the IUSR_<`Server Name`> account, either in the

Windows NT User Manager or in the Internet Service Manager.

IUSR_<Server Name> is an account that is created when you set up IIS for the purpose of allowing anonymous Internet access to resources on the Web server. The IUSR_<Server Name> account is created with a randomly generated password. If the password is changed in the Internet Service Manager, the actual Microsoft Windows NT account password must also be changed in the Microsoft Windows NT User Manager.

If the System DSN that your IDC or ASP files use points to a Microsoft Access database located on a different Microsoft Windows NT computer than your Web server, then the same IUSR_<Server Name> account with the same password must be created on the computer where the Microsoft Access database is located. If your Microsoft Access database is not on a different computer than your Web server, but it uses attached tables that are on a different computer, then the same IUSR_<Server Name> account with the same password must be created on the computer where the attached tables are located. Remember that the IUSR_<Server Name> account is created initially with a randomly generated password; the only way to know the password is to change it to something else.

Insufficient NTFS Directory Permissions

The IUSR_<Server Name> account requires both Read and Write permissions on the directories where your Microsoft Access databases are located. Keep in mind that NTFS directory permissions are different than share permissions. If your databases are on a different computer than your Web server, or if you are using attached tables, then Read and Write permissions must be granted on the directories on both computers where the Microsoft Access databases reside.

Insufficient Share Permissions

If the System DSN points a Microsoft Access database that resides on a different computer than your Web server, then the IUSR_<Server Name> account must exist on the other computer, and should have Read and Write permissions on the share where the Microsoft Access database resides.

Access Database Is on a Windows 95 Computer with Insufficient Share Permissions

If your System DSN points to a Microsoft Access database that is located on a Microsoft Windows 95 computer, whatever account is set up to be the Anonymous Logon in the Internet Service Manager must be authenticated by the share permissions on Microsoft Windows 95. This means if IUSR_<Server Name> is set up as the Anonymous Logon, then IUSR_<Server Name> must also be a valid account in the domain, with the same password used in the Internet Service Manager. The "Username:" value under "Anonymous Logon" in the Internet Service Manager must be modified to use the following format:

DOMAIN\IUSR_<Server Name>

Note The following error message is caused by a Microsoft Access database that resides on a different computer than the Web server, and that has tables linked to a Microsoft Access database on a third computer:

'<path and file name>' isn't a valid path. Make sure that the path name is spelled correctly and that you are connected to the server on which the file resides.

To resolve this error, store the front end database on the same computer as the Web server, or do not use tables linked to a Microsoft Access database that resides on a third computer.

References

For information about exporting IDC files in Microsoft Access 97, search the Microsoft Access 97 Help Index for "IDC files."

For information about exporting ASP files in Microsoft Access 97, search the Microsoft Access 97 Help Index for "ASP files."

For more information about setting Microsoft Windows NT Share permissions and NTFS directory permissions, please refer to your Microsoft Windows NT documentation.

For more information about the Internet Information Server and the Internet Database Connector, please refer to

your IIS online documentation.

KBCategory: kbtshoot kberrmsg
KBSubcategory: IntpOthr EvnOthr
Additional reference words: 97 8.00 troubleshoot tshoot html iis

```
Dim ctxObject As ObjectContext

Set ctxObject = GetObjectContext()
```

```
'This code is placed before the <HTML> tag
'Run code to perform work here
If err.number <> 0 Then
   session("ErrorTitle") = "Error Title to be Displayed"
   session("ErrorText") = "Description of Error"
   'Redirect to custom built error page that will display the session variables
   response.redirect "error.asp"
End If
```

Article ID: Q161172
Creation Date: 17-DEC-1996
Revision Date: 06-FEB-1997

The information in this article applies to:

® Microsoft Access 97

Summary

Advanced: Requires expert coding, interoperability, and multiuser skills.

Warning Although this article discusses Microsoft Access security features, any information you send over the Internet with the techniques described in this article is sent unencrypted. To send encrypted information over the Internet, you must use a protocol that sends client certificates, such as Secure Sockets Layer (SSL). Note that client certificates cannot be used on Personal Web Server for Windows 95. ANY USE BY YOU OF THE METHODS PROVIDED IN THIS ARTICLE IS AT YOUR OWN RISK. Microsoft provides this sample "as is" without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability and/or fitness for a particular purpose.

This article describes a technique you can use to create Internet Database Connector (IDC) files that allow you to type a username and password in an HTML form in order to query a secure Microsoft Access database.

More Information

There are three main steps to use IDC files to query a secure Microsoft Access database:

® Use ODBC Administrator to create a System DSN that points to the workgroup information file (System.mdw) you use with your secured database.

® Create an HTML form that requests a username and password. The HTML form passes the values to parameters in your IDC file.

® Modify the IDC file to use the username and password parameters to authenticate user access to your database.

You must use an HTML form to enter the username and password, and then pass that information to the IDC files. You cannot configure Microsoft Internet Information Server (IIS) Basic Authentication or NT Challenge/Response to achieve this functionality because those IIS options authenticate users against Microsoft Windows NT permissions, not Microsoft Access security accounts. It is possible to use Basic Authentication and NT Challenge/Response with Microsoft SQL Server databases because SQL Server can be integrated with NT Security. Microsoft Access security does not provide that capability.

This example contains the following sections:

® Creating a Secure Copy of Northwind.mdb

® Creating a System DSN for a Secure Microsoft Access Database

® Creating the HTX/IDC Files and the HTML Logon Form

® Customizing the IDC Files

® Testing the Query

Creating a Secure Copy of Northwind.mdb

1. Copy the sample database Northwind.mdb to C:\My Documents\Northwind2.mdb.
2. Start the Workgroup Administrator by executing the Wrkgadm.exe file. The file is installed to your Windows\System folder by default. You can find a shortcut to this file in the C:\Program Files\Microsoft Office folder.
3. In the Workgroup Administrator dialog box, click Create.
4. In the Workgroup Owner Information dialog box type the following information, and then click OK:

   Name: <Your name>

Organization: Northwind Traders
Workgroup ID: 1234

5. In the Workgroup Information File dialog box, make a note of the path and file name that appears in the Database box so you can rejoin that workgroup file when you are finished. Then type C:\My Documents\Northwind2.mdw, and click OK.
6. In the Confirm Workgroup Information dialog box, verify that the information is correct, and then click OK.
7. Click OK when you see the message that you have successfully created the workgroup file, and then click Exit in the Workgroup Administrator dialog box.
8. Start Microsoft Access and open C:\My Documents\Northwind2.mdb.
9. On the Tools menu, point to Security, and then click User And Group Accounts.
10. Click the Change Logon Password tab, and type Admin in the New Password and Verify boxes. Click OK.
11. Quit Microsoft Access.
12. Copy the Northwind2.mdb and Northwind2.mdw files to a folder on your Web Server computer, or to a network location that you can access from your Web Server computer. Be sure to retain a copy of Northwind2.mdb on your local drive for use later in this example. At this point, you can rejoin your original workgroup information file.
13. Start the Workgroup Administrator program following the procedure in step 2 of this section.
14. Click the Join button.
15. In the Workgroup Information File dialog box, type the path and file name of the system database that you noted in step 5, and then click OK.
16. Click OK when you receive the message that you have successfully joined the workgroup, and then click Exit in the Workgroup Administrator dialog box.

Creating a System DSN for a Secure Microsoft Access Database

1. Double-click the ODBC icon in Control Panel on your Web Server.
2. In the ODBC Data Source Administrator dialog box, click the System DSN tab.
3. Click the Add button.
4. Select Microsoft Access Driver, and then click Finish.

Note If the Microsoft Access Driver does not appear, it is not installed on your Web server. For information about installing the driver on your Web server, search the Help Index for "Microsoft Access Desktop driver," or ask the Microsoft Access 97 Office Assistant.

---

5. In the ODBC Microsoft Access 97 Setup dialog box, type NorthwindIDC in the Data Source Name box.
6. Click the Select button and browse to select Northwind2.mdb. Click OK.
7. In the System Database box, click Database, and then click the System Database button. Browse to select Northwind2.mdw, and then click OK.
8. Note that you have the option to click the Advanced button in the ODBC Microsoft Access 97 Setup dialog box, and set a default Login name and Password for the System DSN. Any of your IDC files that do not provide a username and password will use the default settings.
9. Click OK to close the ODBC Microsoft Access 97 Setup dialog box.
10. Click OK to close the ODBC Data Source Administrator dialog box.

Creating the HTX/IDC Files and the HTML Logon Form

In this section, you create a query with username and password parameters, and then export the query to HTX/IDC format. When you create IDC files from a parameter query, Microsoft Access automatically creates an HTML form for entering the parameters. This is an easy way to create the HTML form you need to collect the username and password information. However, you do not have to use this technique to create the HTML form; you can use Notepad or another tool, such as Microsoft Front Page 97, to create your own HTML Logon form.

1. Start Microsoft Access.

2. Open Northwind2.mdb.
3. Create a new query called SecureIDC based on the Customers table:

```
Query: SecureIDC
-------------------
Type: Select Query
Field: CustomerID
   Table: Customers
```

4. On the Query menu, click Parameters.
5. Type the following in the Query Parameters dialog box, and then click OK.

```
Parameter           Data Type
----------------------------
[UserParam]         Text
[PassParam]         Text
```

6. Save the SecureIDC query and close it.
7. Select the SecureIDC query in the Database window, and then click Save As/Export on the File menu.
8. In the Save As dialog box, click "To an External File or Database," and then click OK.
9. In the "Save Query 'SecureIDC' In" dialog box, select Microsoft IIS 1-2 (*.htx;*.idc) in the Save As Type box, and type SecureIDC.htx in the File Name box. Click Export.
10. In the HTX/IDC Output Options dialog box, type NorthwindIDC in the Data Source Name box, and then click OK.
11. Click OK in each of the two Enter Parameter Value dialog boxes that appear.
12. The HTX/IDC output creates three files: SecureIDC.HTML, SecureIDC.htx and SecureIDC.IDC.

Customizing the IDC Files

Note This section contains information about editing IDC and HTML files, and assumes that you are familiar with editing HTML files. Microsoft Technical Support engineers do not support customization of any HTML, HTX, IDC, or ASP files.

---

1. Use Notepad or another text editor to open the SecureIDC.IDC file. The Password and Username fields show nothing entered next to them by default. You must add the parameters that will hold the values from the HTML Logon form. Change the SecureIDC.IDC file so it looks as follows:

```
Datasource: NorthwindIDC
Template:SecureIDC.htx
Username:%[UserParam]%
Password:%[PassParam]%
SQLStatement:SELECT Customers.CustomerID
+FROM Customers;
```

2. Save the SecureIDC.IDC file and close it.
3. Use Notepad or another text editor to open the SecureIDC.HTML file. By default, the HTML form uses the GET method to submit its data. GET variables appear in the address box of Web browsers. Therefore, you must change the GET method to the POST method if you do not want your password to be visible in the address box of your Web browser. Locate the following line in the SecureIDC.HTML file

```
<FORM METHOD="GET" ACTION="SecureIDC.IDC">
```

   and change it to:

```
<FORM METHOD="POST" ACTION="SecureIDC.IDC">
```

4. Text boxes use an Input Type setting of Text by default. In order to prevent your password from being visible in the text box on your form, you must change the Input Type to Password. Locate the following line in the SecureIDC.HTML file

```
[PassParam] <INPUT TYPE="Text" NAME="[PassParam]"><P>
```

and change it to:

```
[PassParam] <INPUT TYPE="Password" NAME="[PassParam]"><P>
```

5. Save the SecureIDC.HTML file and close it.
6. Copy SecureIDC.HTML, SecureIDC.htx and SecureIDC.IDC to a folder on your Web Server computer where you have both Read and Execute permission.

For more information about configuring Microsoft Internet Information Server (IIS) permissions, please refer to the IIS Help Index, and see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q160754**
TITLE: PRB: Error "HTTP/1.0 403 Access Forbidden" Browsing IDC Page

Testing the Query

1. Start Microsoft Internet Explorer 3.0, or another Web browser program.
2. Type the Uniform Resource Locator (URL) in the address box of your Web browser to view SecureIDC.HTML. For example, if you saved your IDC files in a folder called Test in the wwwroot folder of your Web Server, type:

   http://<servername>/test/SecureIDC.HTML

   Note that the URL depends upon where your files are located on the Web Server.
3. The SecureIDC.HTML form opens in your Web browser with a [UserParam] box, a [PassParam] box, and a Run Query button. Type Admin in both boxes, and then click the Run Query button. The SecureIDC.IDC file opens and displays a list of CustomerIDs.

Note If you type an incorrect username or password, you receive the following error:

Error Performing Query
Not a valid account name or password.

References

For more information about how to create and modify the optional fields in IDC files, please refer to your Microsoft IIS online documentation.

For more information about IIS authentication, security, and Secure Sockets Layer (SSL), please refer to your IIS online documentation, or see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q142868**
TITLE: Authentication & Security Features

For more information about Microsoft Access security, search the Help Index for "security, overview," or ask the Microsoft Access 97 Office Assistant.

KBCategory: kbusage kbhowto
KBSubcategory: IntpOthr
Additional reference words: 97 8.00 internet intranet

Article ID: Q160833
Creation Date: 11-DEC-1996
Revision Date: 24-JAN-1997

The information in this article applies to:

® Microsoft FrontPage 97 for Windows

Summary

FrontPage can convert pasted text, inserted file segments in HTML pages, or entire files from several formats to HTML. This article presents an overview of how FrontPage 97 handles document conversion to HTML and answers the following questions:

® When will FrontPage attempt to convert a file to HTML?

® Why does the conversion process show two stages?

® How does FrontPage decide which converter to run?

® What Formats can FrontPage 97 convert to HTML?

® How Are Converters Installed and Where Are They Located?

More Information

When Will FrontPage Attempt to Convert a File to HTML?

FrontPage will attempt to convert a File to HTML when you do any of the following:

® Paste text from a non-HTML document

® Click File on the Insert menu in FrontPage Editor and select a non-HTML document

® Use the right mouse button to drag a non-HTML document to FrontPage Explorer. When you release the mouse button, click "Copy Here as Web Format" from the shortcut menu.

Why Does the Conversion Process Show Two Stages?

FrontPage 97 first converts documents from their native format to Rich Text Format (RTF), and then it converts the RTF files to Hypertext Markup Language (HTML). If the incoming file is already in HTML format or in simple text, no converter is run. If the incoming file is RTF only, the RTF to HTML converter is run.

How Does FrontPage Decide Which Converter to Run?

When FrontPage imports a file, the algorithm for determining which converter will be run is as follows:

First the conversion routine looks at the extension of the file. For each converter registry entry that matches the extension of the file, that converter's entry point for the IsFormatCorrect routine is run. This routine reads the first few bytes of the incoming file and decides if it understands the format. The first converter that understands the format is then run to convert the file to RTF format.

If FrontPage is unable to locate a converter that understands the file based on its extension, this procedure is repeated for each converter registry entry regardless of the file's extension.

If FrontPage still can't find a converter to open the file, it will display the "Open File As" dialog box with buttons you can click to open the file as RTF, HTML, or text.

What Formats Can FrontPage 97 Convert to HTML?

FrontPage 97 installs RTF converters for the following formats:

® Microsoft Excel Worksheet (.xls, .xlw)

® Word (Asian versions) 6.0/95 (.doc)

® Word 2.x for Windows (.doc)

® Word 4.0/5.1 for Macintosh (.mcw)

® Word 6.0/95 for Windows&Macintosh (.doc)

® Word 97 (.doc)

® WordPerfect 6.x (.wps, .doc)

® Works 3.0 for Windows (.wos)

® Works 4.0 for Windows (.wps)

These converters convert from the formats above to RTF format. FrontPage 97 will also install the converter to convert a file from RTF to HTML format:

® HTML Document (.html, .htm, .htx, .asp)

How Are Converters Installed and Where Are They Located?

FrontPage 97 and Office 95 and Office 97 and other Microsoft programs use the same location for shared converters (and the same registry keys to point at these converters), installing or uninstalling any of these programs may change the versions of the converters that you are running.

Note The FrontPage 1.1 "RTF to HTML" converter (rtf2html.dll) is not included with FrontPage 97 for Windows.

FrontPage 97 Setup does not treat converters as shared components. For information about problems that may occur as a result of this behavior, please see the following articles in the Microsoft Knowledge Base:

ARTICLE-ID: **Q160226**
TITLE: Can't Use Proofing Tools (Spelling Checker) or Text Converters

KBCategory: kbusage kbinterop kb3rdparty
KBSubcategory: fphtml
Additional reference words: 97

ID: Q164059
Created: 26-FEB-1997
Modified: 27-FEB-1997

The information in this article applies to:

® Microsoft Internet Information Server, versions 2.0 and 3.0

Symptoms

Note On Sunday, February 23, 1997, Microsoft was alerted to a posting regarding an Internet Information Server (IIS) security exposure. This bug permits the publication of IIS executable files via a complicated string of commands sent from a web browser to an IIS server.

Because of this security exposure, Internet users can view Active Server Pages (ASP), Internet Server API applications (ISAPI), Internet Database Connector (IDC) applications, or Common Gateway Interface (CGI) applications within an IIS publication installation.

Resolution

Get the hotfix mentioned below.

This hotfix has been posted to the following Internet location. You can download any of these self-extracting files from the following service:

Internet (anonymous FTP)
ftp ftp.microsoft.com
Change to the bussys/winnt/winnt-public/fixes/usa/nt40/hotfixes-postSP2/iis-fix/ folder.
Get Readme.1st (for instructions on downloading and installing the hotfix).

Or use the following full URL on your client browser:

FTP://ftp.microsoft.com/bussys/winnt/winnt-public/fixes/usa/nt40/hotfixes-postSP2/iis-fix/readme.1st

Status

Microsoft has confirmed this to be a problem in Microsoft Internet Information Server versions 2.0 and 3.0.

A supported fix is now available, but has not been fully regression-tested and should be applied only to systems experiencing this specific problem. Unless you are severely impacted by this specific problem, Microsoft recommends that you wait for the next Service Pack that contains this fix. Contact Microsoft Technical Support for more information.

KBCategory: kbusage kbbug2.00 kbbug3.00
KBSubcategory: iissecurity iiswww iisapi

Article ID: Q143130
Creation Date: 11-DEC-1996
Revision Date: 13-DEC-1996

The information in this article applies to:

® Microsoft Transaction Server version 1.0

Summary

This article contains the answers to frequently asked questions about Microsoft Transaction Server.

More Information

When a package is exported, are the package properties pre-set when the package is installed on another computer?

Yes. Determining and setting properties is one of the key tasks in package deployment. By pre-setting package properties, the administrator of the package can control how components and users interact with each other.

Can packages include components that update data in sources other than databases?

Yes. Sources of data for package components are known as Resource Managers. Although a broad range of database products can serve as Resource Managers, at this time there is no capability for non-database products (such as a Microsoft Excel spreadsheet) to serve as a Resource Manager. This limitation does not prevent you from building components that use non-database sources, as long as you are aware of the lack of transaction coordination that will occur with these components. This allows you to exploit the Distributed Component Object Model (DCOM) facility of Transaction Server in a non-transactional environment.

Can packages include components that do not support transactions?

Yes. Transaction processing is only one part of the Microsoft Transaction Server programming model. Packages whose objectives do not include transaction processing can still take advantage of DCOM and Transaction Server s process level security system. In this case, Microsoft Transaction Server is being used to manage object instances and object lifetimes. Microsoft Transaction Server can do this even when no transactions are involved.

Applications that access transactional Resource Managers can simultaneously access non-transactional Resource Managers. The non-transactional resources will not be rolled back if the transaction aborts. If there is a requirement for data synchronization, the non-transactional Resource Manager must have some way to cope with this situation. The work a component does is only transaction-protected if the Resource Managers that the component uses are transactional. To support transactions, the Resource Manager engine must support logging (or some equivalent mechanism) to rollback aborted transactions. The Resource Manager engine must also support either OLE transactions or the XA two-phase commit protocol. These two-phase commit protocols permit the Resource Manager engine to coordinate the transaction outcome with the Microsoft Transaction Server Transaction Manager. The "Microsoft Distributed Transaction Coordinator Resource Manager Developer s Guide" describes how an existing Resource Manager can be enhanced to work with OLE Transactions. You can obtain this document from ftp.microsoft.com using anonymous ftp. The Resource Manager Developer s Guide and a sample Resource Manager are located in the Bussys/Viper directory of the ftp site.

The resource Manager client library must provide a means for the Resource Manager engine to be enlisted in a transaction. To achieve the best performance, the client library should be implemented as a resource dispenser. The documentation necessary to develop a resource dispenser will be contained in the Microsoft Transaction Server SDK, available in March 1997.

Can a client component run on a Windows 95 system?

Yes. A client component can run on the Windows 95 operating system. However, the computer running Transaction Server requires Microsoft Windows NT 4.0, which means that all objects that take advantage of transaction support or any of the DCOM features must be located on computers running Windows NT.

Can Transaction Explorer s Security features be accessed programmatically, or are they only set through the Explorer?

All process level security features are set through the Explorer. Component level security can also be implemented by enabling authorization checking on a component.

What determines the privileges that are available for a role?

Privileges are set through the Windows NT security system. As the package administrator, you can establish roles and assign users and groups to that role. Any privileges that have been assigned to users or groups through Windows NT security will also be available through the Transaction Server security system.

If a component is configured as Requires New, does the Transaction Server perform deadlock detection?

Yes. The following are two possible scenarios:

® Exclusive Lock: If the data is exclusively locked by the calling component, and the sub-component tries to update it, the underlying database management system will detect the deadlock and abort the offending component.

® Non-Exclusive Lock: When the sub-component is not trying to update, the blockage will be detected by either the Activity or Transaction timeout.

In either of these scenarios, Transaction Server will ensure that all of the ACID transaction properties are enforced.

Are there two distinct logs maintained for Transaction Errors, one for Transaction Server and another for the Resource Manager?

If the Resource Manager is SQL Server, then there is only one log, because both Transaction Server and SQL Server are using Microsoft Distributed Transaction Coordinator (DTC). If the application is accessing an ODBC Resource Manager, there will be two error logs.

Does Transaction Server run on a computer running Windows NT Workstation, or does it require Windows NT Server?

Transaction Server will run on either system; however, Windows NT Server is recommended as the platform for deploying packages.

Is there a way to view the objects without the hierarchy in the Explorer?

Yes. The hierarchy can be turned off by clicking Hierarchy on the View menu in the Explorer. When the hierarchy is not displayed, you will only see folders and objects that have been added to the run- time environment.

Can I add components with different threading models in the same package?

No. This limitation exists because components with different threading models cannot be part of the same activity or transaction within a single application server process (ASP).

If you need to mix VB and VC components in the same package, you must identify the VC components as single-threaded. You can configure components with different threading models so that they can be part of the same activity, or participate in the same transaction. To do this, add all apartment-thread aware components in one package and all single-threaded components in another package.

Why can the Explorer set the transaction properties to "Does not support transactions?"

This option exists because the person using the Explorer should be the administrator, and this option provides more flexibility. This option allows the ease of implementation for DCOM that the product offers, regardless of the use of transactional resources.

Can I group components that don't have transaction capabilities in a package and use the

Transaction Explorer to set their transaction properties to "requires a transaction," so that they can work in one transaction?

Simply being in a package does not mean that the components will be grouped together in a transaction. This is more of a logical boundary than a physical one. You can only really group the components together from the client through ITransactionContext, or from the component through IObjectContext. Grouping components that are not capable of backing out work they do as transactional will not ensure that work is done properly. Components that update resources must be capable of backing out that update, to ensure data integrity within the transaction.

Is it possible to call CoGetCallContext from a Transaction Server server component, and from that call be able to impersonate the client?

Yes. However, impersonation will reduce the usefulness of connection pooling of ODBC database connections.

KBCategory: kbref kbfaq
KBSubcategory:
Additional reference words: 1.00 faq

Article ID: Q162975
Creation Date: 31-JAN-1997
Revision Date: 31-JAN-1997

The information in this article applies to:

® Microsoft Access 97

Summary

You will not be able to view HTML, IDC, or ASP files exported from Microsoft Access 97 if the required WWW Directory Access permissions are not set on your Internet Server. This article is an overview of the Internet Server WWW Directory Access permissions necessary to view HTML, IDC, and ASP files in a Web browser.

More Information

The following are excerpts from Chapter 5 of the online IIS Installation and Administration Guide which is installed along with IIS:

"Read permission enables Web clients to read or download files stored in a home directory or a virtual directory. If a client sends a request for a file that is in a directory without Read permission, the Web server returns an error. Generally, you should give directories containing information to publish (HTML files, for example) Read permission. You should disable Read permission for directories containing Common Gateway Interface (CGI) applications and Internet Server Application Program Interface (ISAPI) DLLs to prevent clients from downloading the application files."

"Execute permission enables a Web client to run programs and scripts stored in a home directory or a virtual directory. If a client sends a request to run a program or a script in a folder that does not have Execute permission, the Web server returns an error. For security purposes, do not give content folders Execute permission."

Please refer to your online IIS Help Topics for detailed information on configuring Microsoft Internet Information Server (IIS) permissions and securing your Web site.

HTML files require WWW Read permissions. IDC/HTX and ASP files require WWW Execute permissions.

When exporting parameter queries to IDC/HTX or ASP formats, an HTML file is created along with the IDC/HTX or ASP files. All the files may be copied to a folder on your Internet Server that has both Read and Execute permissions. However, for security purposes, it is recommended that IDC/HTX and ASP files be placed in a folder with Execute permission only and that HTML files be placed in a folder with Read permission only.

References

For more information about permissions necessary for IDC/HTX files, please see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q160754**
TITLE: Error "HTTP/1.0 403 Access Forbidden" Browsing IDC Page

For more information about Windows NT permissions, please see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q161333**
TITLE: Check NT Permissions When Using IDC/ASP Files with Access

KBCategory: kbinterop kbhowto
KBSubcategory: IntpOthr WzOthr NpdOthr
Additional reference words: 97 7.00

Article ID: Q161420
Creation Date: 21-DEC-1996
Revision Date: 31-DEC-1996

The information in this article applies to:

® Microsoft FrontPage 97 for Windows

Symptoms

When the FrontPage Editor parses HTML, there are certain tags for which unknown attributes are discarded.

Cause

FrontPage normalizes HTML to the HTML 2.0 specification (RFC 1866), or, where the 2.0 specification is superceded, to the HTML 3.2 proposal.

Workaround

If you want to write HTML with unknown attributes to tags identified in the "More Information" section of this article, put the entire tag into an HTML Markup section.

To insert an HTML Markup section in a FrontPage web page, click HTML Markup on the Insert menu in FrontPage Editor.

Status

Microsoft is researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

More Information

The FrontPage Editor knows about a certain set of attributes for each HTML tag it can write. These known attributes are preserved by the FrontPage Editor, and the user can manipulate them using dialog boxes and direct manipulation in the "View or Edit HTML" dialog box. When the FrontPage Editor encounters an attribute it doesn't recognize, it attempts to put that attribute and its value into the extended attribute list for that tag. Any tag for which the FrontPage Editor preserves extended attributes will include an Extended button in its Properties dialog box. The attribute list is preserved by the FrontPage Editor in its extended attributes list for that tag and is inserted into the final HTML.

Not all tags have extended attribute lists in FrontPage. The following HTML tags don't have an extended attribute list in FrontPage Editor:

AREA, B, BASE, BASEFONT, BGSOUND, BIG, BLINK, CENTER, CITE, CODE, DFN, DIV, EM, FONT, HEAD, HTML, MAP, META, NEXTID, NOFRAMES, OPTION, SAMP, SCRIPT, SMALL, STRIKE, S, STRONG, SUB, SUP, TITLE, TT, U, VAR, NOEMBED, PARAM.

When you work with these tags, unrecognized attributes are discarded by the FrontPage Editor.

KBCategory: kbusage
KBSubcategory:
Additional reference words:

Article ID: Q160754
Creation Date: 09-DEC-1996
Revision Date: 10-DEC-1996

The information in this article applies to:

® Microsoft Access 97 •Microsoft Internet Information Server versions 1.0, 2.0

Symptoms

Moderate: Requires basic macro, coding, and interoperability skills.

When you use a Web browser to open an Internet Database Connector (IDC) file, you may receive one of the following error messages:

> HTTP/1.0 403 Access Forbidden (Execute Access Denied - This Virtual Directory does not allow objects to be executed.)
>
> HTTP/1.0 403 Access Forbidden (Read Access Denied-This Virtual Directory does not allow objects to be read.)

Cause

These errors occur if you do not have Execute or Read permission for the virtual directory on the Internet server in which the IDC file is located.

Resolution

You can relocate the IDC file to a directory on the server where you have Execute or Read permission, or you can follow these steps on the Internet server computer to assign the correct permission for the IDC file's current directory:

1. Start the Microsoft Internet Information Server (IIS) Internet Service Manager program on the Internet server computer.
2. Double-click the computer name associated with the WWW service. The WWW Service Properties dialog box appears.
3. Click the Directories tab.
4. Select the directory in which the IDC file is located, and then click the Edit Properties button.
5. In the Directory Properties dialog box, click Execute, click Read, or click both options, and then click OK.
6. Click OK to close the WWW Service Properties dialog box, and then quit the Internet Service Manager program.

More Information

You must have the correct permission for an Internet server directory that contains IDC files. When IIS receives a URL that contains an .idc file extension, it uses Open Database Connectivity (ODBC) to execute a query against a data source and returns the results to you. If you do not have Execute permission, IIS cannot execute the query. If you do not have Read permission, you cannot open the IDC file.

Steps to Reproduce Behavior

The following steps assume that you have an ODBC System DSN on your Internet server called MySystemDSN.

Setting Permissions on the Internet Server

1. On the Internet server, start the IIS Internet Service Manager.
2. Double-click the computer name associated with the WWW service. The WWW Service Properties dialog box appears.
3. Click the Directories tab.
4. Select the directory that corresponds to the /Scripts alias, and then click the Edit Properties button.
5. In the Directory Properties dialog box, click to clear the Execute and the Read check boxes, and then click

OK.

6. Click OK to close the WWW Service Properties dialog box, and then quit the Internet Service Manager program.

Creating and Browsing the HTX/IDC files

1. Start Microsoft Access 97 and open the sample database Northwind.mdb.
2. Select the Shippers table in the Database window.
3. On the File menu, click Save As/Export.
4. In the Save As dialog box, click "To an External File or Database," and then click OK.
5. In the "Save Table 'Shipper' In dialog box," select Microsoft IIS 1-2 in the Save As Type box, and then click Export.
6. In the HTX/IDC Output Options dialog box, type MySystemDSN in the Data Source Name box. If your System DSN requires it, type your user name in the "User to Connect As" and your password in the "Password for User" boxes. Click OK.
7. Microsoft Access creates a Shippers.htx and a Shippers.IDC file on your computer. Move them to the /Scripts virtual directory on your Internet server computer.
8. Start your Internet browser and type the following URL:

   http://<machine name>/scripts/shippers.idc?

   where <machine name> is the name of the Internet server. Note that you receive the error message:

   HTTP/1.0 403 Access Forbidden (Execute Access Denied-This Virtual Directory does not allow objects to be executed.)

9. Type the following URL in your browser:

   http://<machine name>/scripts/shippers.idc

   where <machine name> is the name of the Internet server. Note that you receive the error message:

   HTTP/1.0 403 Access Forbidden (Read Access Denied-This Virtual Directory does not allow objects to be read.)

References

For more information about creating Web pages in Microsoft Access, search the Help Index for "pages, Web pages," or ask the Microsoft Access 97 Office Assistant.

For more information about virtual directories, search the Microsoft Internet Information Server Help Index for "Virtual directories."

KBCategory: kbenv kbprb
KBSubcategory: EvnNtw IntpOthrAdditional reference words: 97 IIS 8.00

Article ID: Q152828
Creation Date: 25-JUN-1996
Revision Date: 30-AUG-1996

The information in this article applies to:

® Microsoft SQL Server, versions 4.21, 6.0 and 6.5

Summary

When you use anonymous connections for IIS and you attempt to query remote SQL servers using named pipes, the following error may occur:

```
[State =01000][Error=1326][Microsoft][ODBC SQL Server Driver]
[SQL Server][dbnmpntw] ConnectionOpen (CreateFile())
```

Error 1326 means "Logon failure: unknown user name or bad password." This error is raised because the IIS Service does not have any rights to use the named pipe.

More Information

In order for a SQL server client to gain access to a Windows NT named pipe, the client needs to be validated by the Windows NT server. This is normally accomplished through a Workgroups style validation method where identical usernames and passwords are created on the client and the server, or the domain method where both the client and server are domain members.

The Internet Information Server (IIS) setup process creates a Windows NT ID called IUSR_machinename (where machinename is the name of the Windows NT server), and adds that user to the Guest local group. The IIS service runs under this Windows NT account name.

If the connection to IIS is anonymous, IIS uses the Windows NT account IUSR_machinename and so any network activity performed by IIS is done under this account ID.

If the SQL server resides on the same server as IIS, then named pipes connections work and validation is successful.

If the SQL server resides on a separate server from IIS, then the IUSR_machinename account needs to be validated on the Windows NT server that SQL resides on.

This validation can be implemented by several methods:

® Add the account to the local user account database on the NT server that SQL resides on.

® Make the Windows NT ID IUSR_machinename a member of the domain that the SQL server resides in.

® Enable the Windows NT ID Guest.

For more information regarding security and Microsoft Internet Information Server, please refer to the following article in the Microsoft Knowledge Base:

Article-ID: **Q142868**
TITLE: Authentication and Security Features

KBCategory: KBSubcategory:
Additional reference words: ODBC IIS

Article ID: Q142868
Creation Date: 21-JAN-1996
Revision Date: 29-AUG-1996

The information in this article applies to:

® Microsoft Internet Information Server version 1.0

Summary

This article describes the Authentication & Security Features of Microsoft Internet Information Server.

More Information

Integration with Windows NT

The World Wide Web (WWW), Gopher, and FTP services included with the Microsoft Internet Information Server are fully integrated with Windows NT Server user accounts and file access permissions.

Every access to a resource (for example, a file, an HTML page, an Internet Server API (ISAPI) application, etc.) is done by the services on behalf of a Windows NT user. The service impersonates the user by supplying a username/password pair in the attempt to read/execute the resource for the client.

The Windows NT File System (NTFS) allows Access Control Lists (ACLs) to be assigned to files and directories. ACLs grant and/or deny access to the associated file or directory by specific Windows NT user accounts, or groups of users. When an Internet service attempts to read or execute a file on behalf of a client request, the user account offered by the service must have permission, as determined by the ACL associated with the file, to read or execute the file, as appropriate. If the user account does not have permission to access the file, the request fails, and a response is returned, informing the client that access has been denied.

File and directory ACLs are configured using the Windows NT File Manager.

Anonymous Connections

An anonymous connection is processed when a client request does not contain a username and password. This occurs in the following cases:

® An FTP client logs on with the username Anonymous.

® All Gopher requests.

Note A WWW (HTTP) request's headers do not contain a username and password.

Each Internet service maintains a Windows NT username and password to be used for the processing of anonymous requests. When an anonymous request is received, the service impersonates the user configured as the anonymous-logon user. The request will succeed if the anonymous-logon user has permission to access the requested resource, as determined by the resource's ACL. For WWW only, if the user does not have permission to access the resource, the response returned to the client contains a list of supported authentication schemes for gaining access to the resource. See below for more information.

The anonymous-logon user account can be viewed and modified on the Service property page of the Internet Service Manager. Multiple Internet Information Server services running on the same computer can use the same, or different anonymous-logon user accounts. Including the 'anonymous logon' user account in file or directory ACLs allows for precise control of the resources available to 'anonymous' clients.

The anonymous-logon user account specified must be a valid Windows NT user account on the server computer, and the password specified must match the password for this user in the computer's user database. User accounts and passwords are configured using the Windows NT User Manager.

When the Internet Information Server product is installed, Setup creates a user account on the server computer to be used for anonymous connections. The username of this account has the form IUSR_<computer_name>. For example, if the server computer name is WEB1, the username created will be IUSR_WEB1. The same anonymous-logon user account is set up for all Internet Information Server services installed on the computer.

The account is made a member of the computer's Guest group. This will, in most cases, give anonymous client requests access to public content published on the server. Run the Control Panel/Network applet to see the computer name configured for the Windows NT Server computer.

A randomly generated password is created for the IUSR_<computer_name> account. For maximum convenience and security, we suggest that you change the password associated with this account to a password that you will remember, but is not easily guessed. To do this, you must specify the new password for the account in User Manager, and on the Service page of the Internet Service Manager for each Internet Information Server service installed.

When the Internet Information Server is installed on a primary or secondary domain controller, the anonymous-logon user account is created in the user account database of the domain. When Internet Information Server is installed on a domain member-server, or a stand-alone server, the account is created on the local machine.

If Internet Information Server is installed on multiple domain controllers of the same domain, a separate user account is created in the domain user database for each Internet server computer. This does not cause any conflicts since each username is unique, containing the name of the associated computer.

However, you may find it more convenient to create a single anonymous-logon user account in the domain to use for all Internet Information Server domain controllers in the domain. This can simplify administration of ACLs. To do this, follow these steps:

1. In User Manager, create a new anonymous-logon user account in the domain. Be sure that this account is made a member of appropriate groups, given a secure password, and is given the User Right (in the Policies menu) to Log On Locally.
2. On the Service property page of Internet Service Manager, specify the new anonymous-logon username and password. You must do this for each Internet Information Server service running on all primary and secondary domain controllers in the domain.
3. When later installing Internet Information Server on other domain controllers in the domain, be sure to use Internet Service Manager to modify the anonymous-logon username and password to match those created with User Manager. Do this for each Internet Information Server service installed.

Client Requests Containing Credentials

A request containing credentials is one of the following:

® An FTP client logs on with a valid Windows NT username and password.

This requires that the FTP service checkbox labled Allow Only Anonymous Connections be unchecked. WARNING: FTP sends passwords across the network in clear text.

® A WWW (HTTP) request's headers contain a username and password.

This is HTTP basic authentication. WARNING: HTTP basic authentication sends passwords across the network in clear text.

® A WWW browser supports NTLM (Windows NT native) authentication, and an anonymous client request is denied access to a resource.

In this case, the browser automatically sends the Windows client's username and password to the Internet Information Server Web server using the encrypted NTLM protocol. In this release, only the Internet Explorer 2.0 for Windows 95 supports NTLM authentication.

When an Internet Information Server service receives a client request that contains credentials (a username and password), the anonymous-logon user account is not used in processing the request. Instead, the username and password received by the client are used by the service. If the service is not granted permission to access the requested resource while impersonating the specified user, the request fails, and an error notification is returned to the client.

For WWW (HTTP) only, when an anonymous request fails because the anonymous-logon user account does not have permission to access the desired resource, the response to the client indicates which authentication schemes the service supports. This is determined by the configuration of the WWW service authentication features. If the response indicates to the client that the service is configured to support HTTP basic authentication, most Web browsers will pop up a username/password dialog box, and reissue the anonymous request as a request with credentials, including the username and password entered by the user.

If a Web browser supports NTLM authentication, and the Web service is configured to support NTLM

authentication, an anonymous WWW request failing due to permissions, will result in automatic use of the NTLM protocol to send a username and encrypted password from the client to the service. The client request will then be reprocessed, using the client's user information. The user account obtained from the client is that with which the user is logged into the client computer. As this account, including its Windows NT domain, must be a valid account on the Web server machine, NTLM authentication is most useful in an intranet environment, where the client and server machines are in the same, or trusted domains. In this release, Internet Explorer for Windows 95 is the only browser that supports NTLM authentication.

Internet Service Manager Authentication Options

In addition to the anonymous-logon username and password fields, the Internet Service Manager Service property page contains the following authentication options:

For WWW:

® Allow Anonymous.

When this check box is checked, anonymous connections are processed, and the anonymous-logon username/password are used for these connections. When this checkbox is unchecked, all anonymous connections are rejected. In this case, basic or NTLM authentication can be used to access content.

® Basic.

When this check box is checked, the Web service will process requests using basic authentication. WARNING: Basic authentication sends Windows NT usernames and passwords across the network without encryption. This checkbox is unchecked by default for security reasons.

® Windows NT Challenge/Response.

When this check box is checked, the service will honor requests by clients to send user account information using the Windows NT Challenge/Response (NTLM) protocol. This protocol uses encryption for secure transmission of passwords. The NTLM authentication process is initiated automatically as a result of an 'access denied' error on an anonymous client request. Currently, NTLM authentication only works with Internet Explorer 2.0 for Windows 95.

For FTP:

® Allow Anonymous Connections.

When this check box is checked, FTP logons in which the user enters a username of 'anonymous' will be processed. These anonymous connections will be processed on behalf of the Windows NT user account specified on the Service property page. When this check box is unchecked, users will be required to enter valid Windows NT usernames and passwords to log on to the FTP service.

® Allow only anonymous connections.

When this checkbox is checked, user logons with a username other than anonymous will be rejected.

Warning FTP User names and passwords are sent across the network in clear text. When this check box is unchecked, Windows NT passwords will be sent to the server without encryption. This check box is checked by default for security reasons.

Other Authentication Issues

SSL:

SSL is a WWW feature that supports data encryption and server authentication. All data sent to or from the client using SSL is encrypted. If HTTP basic authentication is used in conjunction with SSL, the username and password are transmitted after being encrypted by the client's SSL support. In this release, the only Web browser that supports SSL is Internet Explorer 2.0, for Windows 95. See the Installation and Planning Guide and online help for more information on SSL.

'Interactive' and 'Network' Users

If you use the predefined Windows NT user accounts 'INTERACTIVE' and 'NETWORK' for access control, your use of these accounts may affect client access to some resources. In order for a file to be accessed by anonymous client requests or client requests using basic authentication, the requested file must be accessible by the INTERACTIVE user. In order for a file to be accessible by a client request using NTLM authentication, the

file must be accessible by the NETWORK user.

Log On Locally User Right

In User Manager, when configuring a Windows NT user account to be used either as the Internet Information Server anonymous-logon account, or as a user account specified by client requests using HTTP basic authentication, be sure that the user account is granted the 'Log on locally' user right. This is specified in User Manager's Policies menu.

Customized Authentication

If you need a WWW request authentication scheme not supported by the service directly, obtain a copy of the Internet Server API (ISAPI) Software Developer's Kit (SDK), and read the ISAPI Filters specification on how to develop user-written ISAPI Filter DLLs that handle request authentication.

KBCategory: kbusage
KBSubcategory: iissecurity
Additional reference words: prodiis 1.00 iis

Article ID: Q119591
Creation Date: 25-DEC-1995
Revision Date: 21-FEB-1997

Summary

This article tells you how to obtain Microsoft drivers, patches, Application Notes, and other support files by downloading them from various online information services. This article contains information on the following topics:

® General Instructions for Downloading

MS-DOS or Windows-Based Files (.exe)

Macintosh-Based Files (.hqx or .sea)

® Specific Instructions for Online Information Services

World Wide Web (WWW) on the Internet

Microsoft Anonymous ftp Server on the Internet

Microsoft Download Service (MSDL)

General Instructions for Downloading

The files Microsoft posts on online services are usually compressed, self- extracting files.

Note Small files may not be compressed. These files will appear with their original extensions and are not self-extracting.

MS-DOS or Windows-Based Files (.exe)

If a compressed, self-extracting file is MS-DOS based or Windows based, it has an .exe extension. These files were compressed using the PKWare file- compression utility. To correctly download and extract an .exe file, do the following:

1. Locate the file you want to download using one of the methods described in the sections in the "Specific Instructions for Online Information Services" section of this article.
2. If you are downloading to a floppy disk, you need a formatted, blank disk. If you are downloading to your hard disk, create a new folder in which you can temporarily place the file and extract it.

   Caution Do not download files directly into your Windows folder. Doing so could overwrite files essential to the proper operation of your system.

   ---

3. Follow the downloading procedure used by your service to retrieve the file you identified in step 1. Download it to a floppy disk or to a new folder.
4. To extract the contents of the self-extracting .exe file after you download it, type the following from the command prompt or the Run command

   ```
   <path\filename> -d
   ```

   where <`path\filename`> is the location and name of the downloaded file. For example, if you downloaded Sample.exe to C:\Download, type the following at the Run command:

   ```
   c:\download\sample -d
   ```

   Note Although you can double-click a self-extracting .exe file to extract its contents, this step provides a method that ensures a recursively compressed file will maintain its file structure.

If you have problems extracting downloaded files, try downloading them again.

Macintosh-Based Files (.hqx or .sea)

If a file is Macintosh based, it has an .hqx extension (except on MSDL, where some files have .sea extensions). An .hqx file is a BinHex 4.0, text- formatted file. After you download the .hqx file and copy it to your Macintosh, decode it with BinHex 4.0 (or another utility that understands BinHex 4 format, such as Stuffit or CompactPro). Decoding converts the .hqx file into a binary-formatted, compressed .sea file. To extract the contents of the file, double-click the <Filename>.sea icon.

Specific Instructions for Online Information Services

World Wide Web (WWW) on the Internet

1. Connect to the Microsoft Web Page at the following address:

   http://www.microsoft.com/

2. Click Support.
3. Click Search the Knowledge Base.
4. In box 1, select "Any Product."
5. In box 2, type "kbfile and <FILENAME>.exe" (include the word "and" but do not include the quotation marks).
6. Click Next.
7. Click the article title to open it and read the complete description of the file.
8. When you're ready to download the file, locate and click file name to download the file.

Note You may need to scroll the article to find the file name. 9.In the dialog box that appears, click Save As. Select the destination folder (the floppy disk drive or the folder you want to download the file to) and click Save.

Microsoft Anonymous ftp Server on the Internet

Type the following at the ">" prompt:

ftp ftp.microsoft.com

Change to the Softlib/Mslfiles folder. To download a file, type the following at the ">" prompt

Get <FILENAME.EXT>

where <FILENAME.EXT> is the name of the file you want to download.

Microsoft Download Service (MSDL)

Using your modem, dial-up (425) 936-6735. This service is available from 1 AM to midnight, 7 days a week. The highest download speed available is 14,400 bits per second (bps). For complete information about using the MSDL, obtain the Application Note GA0604, "Microsoft Download Service (MSDL) Instructions" from online services, or see the following Microsoft Knowledge Base article:

ARTICLE-ID: **Q85774**
TITLE: Instructions for Using the Microsoft Download Service (MSDL)

KBCategory: kbref
KBSubcategory:
Additional reference words: waddle wdl windows driver library softlib data library

Wondering how Visual InterDev can simplify Web development? Take an online test drive and see for yourself. During this tutorial, you'll construct a simple Active Server intranet application for Friendship Insurance, a fictitious insurance company.

The Friendship Insurance intranet application you build will include:

® A dynamically generated HTML report with a listing of current Friendship Insurance customer records (stored in a Microsoft SQL Server or Microsoft Access customer database).

® An HTML data form for insurance agents to directly edit customer records in the database.

® A schedule tracking page that allows agents to track their appointment schedules from their Web browser.

Tutorial.exe is a self-extracting file which includes the sample database and files you'll need to create the Friendship Insurance intranet Web application. Click here to automatically launch tutorial.exe from the \SampApps\Tutorial folder on the *Mastering Web Site Development* CD-ROM.
{ewc mvimg, mvimage,!exec.bmp}

Important You must have the Visual InterDev Beta already installed on your machine before using the tutorial. If you haven't installed a copy of the Beta yet, visit our Beta Download page.

Click here to review the complete instructions for this tutorial, on the Visual InterDev Web site.
{ewc mvimg, mvimage,!intjump.bmp}

For more information about Microsoft Visual InterDev, check out the following articles in the Technical White Papers section of the Library:

® Microsoft Visual InterDev

® Microsoft Visual InterDev Frequently Asked Questions

® Microsoft Visual InterDev Reviewers Guide

Article ID: Q94671
Creation Date: 12-JAN-1993
Revision Date: 03-OCT-1996

To categorize articles within the Microsoft Knowledge Base (KB) and to make finding information easier, Microsoft has developed a common set of keywords for use throughout the KB.

Each KB article has at least one subject keyword, and the first keyword in the list is the primary subject category for that article. There may be additional secondary subject keywords on that same line as well as type keywords (such as kbhowto and kbfile) that categorize the type of the article (such as a how-to article or an article that points to a file).

Note The kbfile type keyword is required in all articles that point to a file in the Microsoft Software Library (MSL) in addition to the primary subject keyword.

At the end of this article is a table that lists formerly used keywords and their replacements.

Categories and Keywords

Keyword	Article Subject
kb3rdparty	Interactions with third-party products
kbdisplay	Display (video, monitor, resolution) issues
kbenv	Environment and configuration information including operating system settings, Windows registry settings, .ini file settings, and development environment issues
bkgraphic	Graphics, including programming issues
kbhw	Hardware
kbinterop	Interaction (connectivity) between Microsoft products (for example, Visual Basic, Microsoft Access, OLE, and MAPI) but not between shared components such as the Jet engine, forms, and so on
kbmm	Multimedia (all homemm articles, Haunted House, Magic School Bus, Explorapedia, and so on), including programming issues
kbnetwork	Networking, including programming issues
kbole	OLE technology, including programming issues
kbother	Other, any subject not covered in other categories
kbpolicy	Support boundaries, policies, processes, and procedures
kbprg	General programming not covered by other subject keywords
kbprint	Printing, including programming issues
kbreadme	Readme files
kbref	Lists (vendor phone numbers, disk directories, reading lists, and so on) FAQs, and other general references
kbsetup	Setup and installation issues
kbsound	Sound (audio) issues
kbtool	Tools, utilities, wizards, compilers, and applets such as MS Draw or Write
kbui	User interface, including programming the user interface
kbusage	Description of product features or functionality
kbappnote	Application notes
kbbuglist	Goes in the official bug list for a particular version of a product
kbbug	Any article that discusses a confirmed bug
kbcode	Sample code

kbdocerr	Documentation errors
kberrmsg	Error message follow-up
kbfaq	List of frequently-asked questions and their answers
kbfasttip	FastTip scripts or maps
kbfile	Pointer to one or more files in the Microsoft Software Library
kbfix	Any article that discusses a fixed confirmed bug
kbfixlist	The official fix list for particular version of a product
kbhelpfile	Article that points to a KB help file
bkhowto	Step-by-step examples or procedures to accomplish a task
kbkeyword	Articles listing product-specific keywords
kbmacro	Sample macro
kbprb	Problems not classified as bugs
kbtlc	Articles and Application Notes that are listed in the FastTip Technical Library Catalog
kbtshoot	Troubleshooting information

KBCategory: kbref
KBSubcategory:
Additional reference words: kbkeyword key word kbcdg dskbguide kbquery

Article ID: Q162977
Creation Date: 31-JAN-1997
Revision Date: 06-FEB-1997

The information in this article applies to:

® Microsoft Access 97

Symptoms

Advanced: Requires expert coding, interoperability, and multiuser skills.

When you browse to an Active Server Pages (ASP) file that was exported from Microsoft Access 97, the Web browser either returns no records or you receive the following error message:

> Expression cannot be used with the LIKE predicate in query expression.

Cause

Because ASP files use ADO to communicate with ODBC drivers in order to query the back-end data, the SQL statements they contain have different character requirements than typical Microsoft Access SQL statements. ASP files use the percent sign (%) as a wildcard character in SQL statements whereas Microsoft Access uses the asterisk (*).

In addition, when Microsoft Access exports a query that contains a parameter concatenated with a wildcard, the SQL statement that is generated does not contain the correct sequence and number of delimiters around the wildcard.

Resolution

Edit the SQL statements in the ASP files so that they use the percent sign (%) for wildcards.

If parameter queries with wildcards are exported, edit the SQL statements in the ASP file so that they use the percent sign (%) as the wildcard along with the correct sequence and number of delimiters around the wildcard.

More Information

Note This section contains information about editing ASP files, and assumes that you are familiar with editing ASP files. Microsoft Access Technical Support engineers do not support customization of any HTML, HTX, IDC, or ASP files.

The following example demonstrates how to change the SQL Statement in an ASP file so that it contains the appropriate wildcard character and the correct sequence and number of delimiters around the wildcard.

1. In Microsoft Access 97, open the sample database Northwind.
2. Create the following new query called FindName based on the Customers table:

```
Query: FindName
----------------------------------
Type: Select Query
Field: CompanyName
   Table: Customers
   Criteria: Like [EnterName] & "*"
```

3. On the Query menu, click Parameters.
4. Type the following in the Query Parameters dialog box, and then click OK:

```
Parameter          Data Type
---------------------------
[EnterName]        Text
```

5. Save the FindName query and close it.
6. Select the FindName query in the Database window, and then click Save As/Export on the File menu.
7. In the Save As dialog box, click "To an External File or Database," and then click OK.
8. In the "Save Query 'FindName' In" dialog box, select Microsoft Active Server Pages (*.asp) in the Save As Type box, and type FindName.asp in the File Name box. Note the folder where the files will be exported to. Click Export. The Microsoft Active Server Pages Output Options dialog box appears.
9. In the Data Source Name box, enter the name of a System DSN that points to the sample database Northwind.mdb.

   For more information on how to define a system DSN, search the Help index for "ODBC, setting up data sources," and see the following article in the Microsoft Knowledge Base:

   ARTICLE-ID: **Q159682**
   TITLE: "Data Source Name Not Found" Err Msg Opening Web Page
10. In the Server URL box, enter the URL that points to the Web Server location where your ASP files will be stored. For example, if you store the ASP files in the \ASPsamp folder on the \\PubTest server, type http://pubtest/aspsamp/ as your Server URL. Click OK.
11. Click OK in the Enter Parameter Value dialog box that appears. Note that the ASP output creates two files: FindName.HTML and FindName.ASP.
12. Copy FindName.HTML and FindName.ASP to a folder on your Web Server computer where you have both Read and Execute permission. Read permission is necessary to browse the HTML file, and execute permission is necessary to run the IDC file.

    For more information about configuring Microsoft Internet Information Server (IIS) permissions, please refer to the IIS Help Index, and see the following article in the Microsoft Knowledge Base:

    ARTICLE-ID: **Q160754**
    TITLE: Error "HTTP/1.0 403 Access Forbidden" Browsing IDC Page
13. Use Notepad or another text editor to open the FindName.ASP file. You need to change the SQL Statement so that it will use the appropriate sequence of parameter and wildcard characters. Change the SQL Statement so that it looks as follows:

    ```
    sql = "SELECT Customers.CompanyName From Customers WHERE (((Customers.CompanyName) Like '" & Request.QueryString("[EnterName]") & "%'))"
    ```

    Note that the wildcard is a percent sign (%) sign and the sequence in which the delimiters were concatenated has changed.
14. Save the FindName.ASP file and close it.
15. Start Microsoft Internet Explorer 3.0 or another Web browser program.
16. Type the Uniform Resource Locator (URL) in the address box of your Web browser to view FindName.ASP. For example, if you saved your ASP files in a folder called Test in the wwwroot folder of your Web Server, type:

    http://<servername>/test/FindName.ASP

    Note that the URL depends upon where your files are located on the Web Server.
17. The FindName.HTML form opens in your Web browser with an [EnterName] box and a Run Query button. Type the letter "M" (without the quotation marks) in the box, and then click the Run Query button. Note that all records with names that begin with "M" are returned.

References

For more information about exporting ASP files, search the Help Index for "ASP files," or ask the Microsoft Access 97 Office Assistant. In addition, please refer to your ASP online documentation.

KBCategory: kbinterop kberrmsg
KBSubcategory:
Additional reference words: 97

Article ID: Q164004
Creation Date: 20-FEB-1997
Revision Date: 21-FEB-1997

The information in this article applies to:

® Microsoft Access 97

® Microsoft Internet Explorer versions 3.0, 3.01 for Windows 95

Symptoms

Moderate: Requires basic macro, coding, and interoperability skills.

When you try to open a Microsoft Access form with a subform in ASP format in Microsoft Internet Explorer, you may receive the following error message

> This page contains active content that is not verifiably safe to display. To protect your computer, the content will not be displayed.

followed by:

> Microsoft VBScript runtime error [Line: NN] Object required: '<`Subform Name`>'

Cause

Your Microsoft Internet Explorer Safety Level is set to High. Internet Explorer opens a second instance in order to display a subform. Internet Explorer itself does not possess a digital security code that marks it "Safe for Scripting." Therefore, when it tries to open the subform in another instance, the error message appears.

Resolution

There are two ways to resolve this problem. You can download the latest HTML Layout Control, or you can set the Internet Explorer Safety Level to Medium or None.

Method 1: Download the Latest HTML Layout Control

There is no security problem with Version 1.0.05.0000 of the HTML Layout Control. Follow these steps to determine which version of the HTML Layout Control you have:

1. Use Windows Explorer to locate the file Isctrls.ocx in your \Windows\System folder.
2. Using the right mouse button, click the file, and then click Properties on the menu that appears
3. Click the Version tab to display the version information.

You can download an updated HTML Layout Control from http://www.microsoft.com/msdownload/ieplatform/iewin95.htm.

You can also download the HTML Layout Control along with the Active X Control Pad from http://www.microsoft.com/workshop/author/cpad/.

Method 2: Change the Microsoft Internet Explorer Safety Level Setting

Set the Internet Explorer Safety Level to Medium or None. Note that if you change the setting to Medium security, you still receive an ActiveX warning message when you try to move between records on a form.

1. Start Microsoft Internet Explorer.
2. On the View menu, click Options.
3. Click the Security tab.
4. Click the Safety Level button.
5. Click either Medium or None in the Safety Level dialog box, and then click OK.
6. Click OK again to close the Options dialog box.

References

For more information about exporting ASP files, search the Help Index for "ASP files," or ask the Microsoft Access 97 Office Assistant.

KBCategory: kbinterop kberrmsg
KBSubcategory: IntpOthr
Additional reference words: 3.00 3.01 97 8.00 IE security

Article ID: Q163948
Creation Date: 20-FEB-1997
Revision Date: 20-FEB-1997

The information in this article applies to:

® Microsoft Access 97

Symptoms

Moderate: Requires basic macro, coding, and interoperability skills.

When you try to enter or change data on an ASP file that is linked to a Microsoft Access database, the new data will be committed only for fields that do not have spaces in their field names. You will not receive any errors about the data not being saved.

Resolution

In order to have the data saved back to a Microsoft Access database, make sure that the underlying recordset does not contain field names that have space in them. Two possible solutions are as follows:

® Redesign your table to include only field names that do not have spaces, and then recreate your ASP files.

Note If you select this option, be sure to update all of the other objects in the database which use the newly modified fields. Otherwise, you will receive errors when you try use those objects.

® Build a query based on the original ASP record source and alias the fields in the query that contain spaces. Once the new query is created, you can recreate your ASP file based on the new query.

Status

Microsoft has confirmed this to be a problem in Microsoft Access 97. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

More Information

Steps to Reproduce Problem

1. Create the following new table called Table1:

```
Table: Table1
------------------------------
Field Name: ID
   Data Type: Counter
   Indexed: Yes (No Duplicates)
Field Name: FirstName
   Data Type: Text
Field Name: Last Name
   Data Type: Text
```

2. Create an AutoForm based on this table.
3. Use the "Publish to the Web Wizard" to create an ASP File. Please note this must be publish to a site which supports ASP file formats and has a system DSN assigned to your database.
4. Use a Web browser to open your new ASP file.
5. Enter and save data in your ASP form.
6. Check the underlining data in the table.

Note that the data you entered into the FirstName field is saved, but that the data you entered into the Last Name field is not saved.

References

For more information about aliasing fields, search the Help Index for "alias," or ask the Microsoft Access 97 Office Assistant.

KBCategory: kbinterop
KBSubcategory: GnlDe
Additional reference words: 97 missing gone saved maintained

Article ID: Q162981
Creation Date: 31-JAN-1997
Revision Date: 31-JAN-1997

The information in this article applies to:

® Microsoft Access 97

Symptoms

Moderate: Requires basic macro, coding, and interoperability skills.

When you export tables, queries, or forms that contain OrderBy and Filter properties to ASP or IDC format, and then browse the exported files in a Web browser, the recordset is not sorted and it outputs all records.

Cause

The Filter and OrderBy properties are not used when the SELECT statements that the Web files use are generated.

Resolution

If you are exporting a table or query, create a new query that implements sorting and criteria in the QBE grid rather than using the OrderBy and Filter properties. Export the new query.

If you are exporting a form, create a new query that implements sorting and criteria in the QBE grid rather than using the OrderBy and Filter properties of the form. Export a form that is based on the new query.

More Information

Please note that IDC files will only display forms in Datasheet view whereas ASP files have the capability to display forms in single Form view. For the system requirements to run IDC or ASP files, please refer to the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q159325**
TITLE: Server and Browser Requirements for Publish to Web Wizard

Steps to Reproduce Behavior

Caution Following the steps in this example will modify the sample database Northwind.mdb. You may want to back up the Northwind.mdb file and perform these steps on a copy of the database.

The following example reproduces the behavior when exporting to IDC. The same behavior will occur when exporting to ASP.

1. In Microsoft Access 97, open the sample database Northwind.mdb.
2. Open the Employees table.
3. Click the First Name column to select it.
4. Click the Sort Ascending button on the toolbar.
5. Close the table and save the changes. The OrderBy property should now contain a string appropriate for the sort that was applied.
6. With the table selected, on the File menu, click Save As/Export.
7. In the Save As dialog box, click to select "To an External File or Database," and then click OK.
8. In the Save As Type box, select Microsoft IIS 1-2(*.htx; *.idc).
9. In the File Name box type "Employees.htx" (without the quotation marks). Note the folder where the IDC/HTX files will be exported to. Make sure the file is exported to a folder with IIS execute permissions.

   For more information about configuring Microsoft Internet Information Server (IIS) permissions, please refer to the IIS Help Index, and see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: **Q160754**
TITLE: Error "HTTP/1.0 403 Access Forbidden" Browsing IDC Page

10. Click Export. The HTX/IDC Output Options dialog box appears.
11. In the Data Source Name box, enter the name of a System DSN that points to the sample database Northwind.mdb. Click OK. Note that two files are created: Employees.idc and Employees.htx.

    For more information about how to define a system DSN, search the Help Index for "ODBC, setting up data sources," and see the following article in the Microsoft Knowledge Base:

    ARTICLE-ID: **Q159682**
    TITLE: "Data Source Name Not Found" Err Msg Opening Web Page
12. Type the Uniform Resource Locator (URL) in the address box of your Web browser to view Employees.IDC. For example, if you saved your IDC/HTX files in the scripts directory of your Web Server, type:

    http://<servername>/Scripts/Employees.idc?

    Note that the URL depends upon where your files are located on the Web Server. The records displayed in the Web browser are not sorted by first name.

References

For more information about exporting IDC files, search the Help Index for "IDC files," and then "Export a datasheet to dynamic HTML format," or ask the Microsoft Access 97 Office Assistant.

For more information about exporting ASP files, search the Help Index for "ASP files," and then "Export a form to dynamic HTML format," or ask the Microsoft Access 97 Office Assistant.

For more information about creating queries, search the Help Index for "queries, creating" and "queries, criteria, entering", or ask the Microsoft Access 97 Office Assistant.

KBCategory: kbinterop
KBSubcategory:
Additional reference words: 97 Internet

What is Microsoft Visual InterDev?

Visual InterDev is a new Web application development tool from Microsoft. Visual InterDev enables developers to easily build dynamic, database-driven Web sites for corporate intranets and the Internet. The tool is a new member of the Microsoft visual tools family.

How does Microsoft Visual InterDev relate to Microsoft FrontPage 97?

Visual InterDev is a Web development tool designed for programmers, while Microsoft FrontPage 97 is a Web authoring tool designed for non-programmers. Microsoft FrontPage 97 is a member of the Microsoft Office family, and looks and works like other Office applications. Visual InterDev is a member of the Microsoft visual tools family, and looks and works like other Microsoft visual development tools (including Visual C++, Visual J++, Visual FoxPro , and Visual Basic). Because most Web sites are created by teams of people, including both non-programmers and programmers, Visual InterDev and Microsoft FrontPage 97 interoperate so that teams of people with different skillsets can work together on the same Web site.

What new features does Visual InterDev offer?

The product includes a variety of visual development tools presented within a single, integrated development environment (IDE). Visual InterDev includes extensive support for building Web applications based on the *Active Server Pages* — an enhancement to the Microsoft Internet Information Server that allows developers to build sophisticated Web applications. Visual InterDev provides a variety of integrated database development tools for building Web sites that are database-driven. Visual InterDev also provides integrated scripting support (VB Script and JavaScript) and enables developers to easily integrate into a Web site both client and server-side components written in Visual Basic, Visual C++, Visual J++ (Java), and other languages.

Is Visual InterDev simply a high-end version of Microsoft FrontPage?

No. Visual InterDev and Microsoft FrontPage are designed for different users. FrontPage 97 is designed to enable *non-programmers*, including business professionals, designers and Web site administrators to easily build and maintain professional Web sites. Visual InterDev is designed to enable *application developers* (programmers) to easily create Web applications, including Web sites with advanced processing such as database connectivity. The tools fully interoperate, so that teams of programmers and non-programmers can work together on the same Web site, each using a tool tailored to their specific needs.

How does Visual InterDev relate to Microsoft Visual Basic, Visual C++ and Visual J++?

Visual InterDev is a member of the visual tools family, and makes it easy to assemble and integrate components developed in the other visual tools within an overall Web solution. Visual J++, Visual C++ and Visual Basic are programming languages that are *component producers*, while Visual InterDev is a c*omponent consumer*. For example, a developer may build a financial analysis component in Visual Basic, and use Visual InterDev to integrate that component into a financial services Web site using Active Server Pages and server-side scripting. The Web site can now provide interactive financial modeling information to users, and since the component is a server-side component, the site can be accessed by any browser on any platform. Besides making it easy to integrate components developed in Visual Basic, Visual C++ and Visual J++ within a Web application, Visual InterDev also shares the same IDE as Visual J++ and Visual C++, providing even tighter integration with these tools.

How does Microsoft Visual InterDev relate to Java?

Visual InterDev is an integrated development environment for building Web applications for corporate intranets and the Internet. Java is a cross-platform programming language for building Web applets and components that can run on either the Web client (inside the browser) or on the Web server. Developers can use Visual InterDev to integrate both client-side Java applets and server-side Java components into a Web site.

Does Visual InterDev work with third-party tools?

Yes. While Visual InterDev includes a comprehensive set of tools for developing dynamic Web sites, customers can easily use Visual InterDev in conjunction with their favorite tools they are already using. Third-party editing tools (such as HTML editors, graphics editors and others) can be easily activated to edit any file that is part of a Visual InterDev project. Additionally, developers can easily integrate components developed in third-party

development tools such as Borland Delphi, MicroFocus COBOL, Sybase PowerBuilder, and many others. Visual InterDev also supports connectivity to ODBC-compliant database system, which includes Oracle, IBM DB/2, and a wide variety of other database systems.

Does Visual InterDev provide database support, and what database systems can be used with Visual InterDev?

Visual InterDev offers sophisticated database development features for building dynamic, data-driven Web sites. Visual InterDev database features are based on Open Database Connectivity (ODBC), and work with most popular database management systems on the market including Microsoft SQL Server , Microsoft Access, Microsoft FoxPro, Oracle, and others.

In addition to providing database connectivity and rich development tools for ODBC-based databases, Visual InterDev also provides a special *Database Designer* feature that brings the ease of use of Microsoft Access to the setup and administration of Microsoft SQL Server databases.

What Web Servers are Supported?

Visual InterDev offers different features based on the capabilities of the Web server you are using. Visual InterDev enables you to create Web projects on Web servers supporting the Front Page Server Extensions, including most popular Web servers on Windows NT and Unix. For example, you can use Visual InterDev's drag and drop publishing features, HTML and client-side development features, and team development features on Microsoft Internet Information Server, Netscape, Apache and other Web servers supported by the Front Page Server Extensions. The core server-side development features such as dynamic HTML generation and database connectivity are based on Active Server Pages, an exciting new development feature for the Microsoft Web servers including Microsoft Internet Information Server 3.0, Peer Web Services for Windows NT Workstation, and the Personal Web Server for Windows 95.

What is an ActiveX Server Component?

An ActiveX Server component (formerly known as an “OLE Automation Server”) is a component that supports the Component Object Model (COM), and executes on a server as opposed to a client. ActiveX Server Components are reusable components that can provide sophisticated functionality. As COM components they expose their functionality to higher-level scripting languages such as VBScript and JavaScript. Thus, ActiveX Server Components are an ideal way to easily provide interactive, dynamic functionality within Web sites using Visual InterDev. ActiveX Server Components can be developed in any language supporting COM, including Microsoft Visual Basic, Visual C++, Visual J++ (Java), and third-party programming languages including Borland Delphi, MicroFocus COBOL and others. ActiveX Server Components also make it easy to integrate existing internal systems with an Internet or intranet-based application, since legacy systems can be “wrapped” as COM components and then integrated into a Web application using Visual InterDev.

Can ActiveX Server Components be distributed on a network?

Yes. The Distributed COM (DCOM) technology for Windows NT and Windows 95 allows any ActiveX Server Component to be seamlessly distributed to any computer on the network. This means that using Visual InterDev, developers can easily build fully distributed, multitier Web applications. By distributing components that require heavy processing to separate application servers, the load on the Web server is reduced, so that greater performance and throughput can be achieved. Distributed solutions can also offer greater fault tolerance.

What are Active Server Pages?

Active Server Pages are a new feature of the Microsoft Internet Information Server 3.0 which makes it easy to develop dynamic, interactive Web sites. Active Server Pages can contain embedded script logic that executes on the server as opposed to the client. Internet Information Server 3.0 provides native support for both VBScript and JavaScript, and third-party scripting engines supporting other scripting languages (such as PERL and others) can be seamlessly plugged in. Active Server Pages can make use of server-side components written in any language supporting the Component Object Model (COM), such as Visual Basic, Visual C++, Visual J++ (Java), COBOL, PASCAL, and other languages.

How Does Visual InterDev support the development of Active Server Pages?

The Visual InterDev integrated development environment (IDE) provides visual tools that automatically generate

much of the Active Server scripting necessary to create dynamic, database-driven Web sites. For example, developers can easily insert database connections into a project to any ODBC data source, and Visual InterDev" will generate all of the Active Server logic necessary to create a pooled database connection to that database for use throughout the Web site. In addition, developers can work with integrated database tools such as the Query Builder, database design-time ActiveX Controls, and database Wizards that make it easy to create Active Server Pages.

What are design-time ActiveX Controls?

Design-time ActiveX Controls are an important new feature introduced with Visual InterDev." Design-time ActiveX Controls provide all of the benefits of component software as standard ActiveX Controls, such as plug-and-play functionality and visual editing at design time. However, design-time ActiveX Controls generate textual content that can include HTML and scripting, viewable on any browser and any platform. They are thus helper components that can enable a developer to visually construct sophisticated Web pages, with the control doing the hard work of generating the HTML and/or scripting necessary to actually build the page. Custom design-time ActiveX Controls provided by third party software vendors enable Visual InterDev to be seamlessly extended with specialized capabilities.

What design-time ActiveX Controls are included with Visual InterDev?

Visual InterDev includes a number of design-time ActiveX Controls, including controls that automate the process of creating dynamic HTML pages based on connections to databases. For example, the Data Command Control provides an easy-to-use visual interface for constructing a complex SQL query against any ODBC database. The control does the hard work of generating the Active Server Page logic (server-side scripting) to execute the query and return a result set for display in a Web page. Other controls include the Form Control that generates databound HTML forms; a Data Range Header Control that automatically creates the server-side scripting for building a page that allows users to page forward and back through a long list of database records; and various other design-time controls that automate the generation of various HTML elements. In addition to these included design-time controls, Visual InterDev also includes over 15 standard ActiveX Controls.

Can third parties build design-time ActiveX Controls?

Yes. Microsoft has held design-previews with independent software vendors, and published the specification for building design-time ActiveX Controls. The specification also includes the information necessary to build tools that make use of design-time ActiveX Controls. We expect that over time, hundreds of design-time ActiveX Controls will be available from third party software vendors, providing a rich set of components that can be used to seamlessly extend the capabilities of Visual InterDev. Design-time ActiveX Controls can be developed in C, Visual C++, and the Visual Basic 5 Control Creation Kit that can be downloaded for free from the Microsoft Web site. See the ActiveX SDK at http://www.microsoft.com/intdev/sdk/dtctrl/ for the complete SDK and source code for several sample design-time controls created in both Visual Basic and Visual C++.

Do design-time ActiveX Controls replace standard ActiveX Controls?

No. Because design-time ActiveX Controls differ from standard ActiveX Controls in that they do not contain a binary, run-time component, Microsoft has decided to name them as a separate category of ActiveX Controls. It is important to note that design-time ActiveX Controls do implement COM interfaces, so they can be arbitrarily shared across development tools provided by various software vendors. However, because they generate HTML and text-based scripting, there is no binary run-time component — hence their output can be viewed on any platform in any browser. These controls are based on published COM interfaces developed during design previews held at Microsoft, and any interested third party can build them. Design-time ActiveX Controls will co-exist with standard ActiveX Controls, which do provide a binary run-time component that offers run-time functionality not offered by design-time ActiveX Controls. For example, standard ActiveX Controls, unlike design-time ActiveX Controls, are exposed as COM objects at run time and design time, and can be scripted at run time by exposed methods, properties, and events.

How does Visual InterDev relate to the Visual Basic 5 Control Creation Kit?

The Visual Basic 5 Control Creation Kit makes it easy to build new components that extend Visual InterDev with new capabilities. Visual Basic developers can now use the Control Creation Kit to build ActiveX Controls, including design-time ActiveX Controls, that can be plugged into Visual InterDev. For example, a developer could use the Visual Basic Control Creation Kit to build a design-time HTML table control that can automatically display a list of database fields in an HTML table element. Such an example, written in VB5, is provided with the

design-time control SDK on the Web at http://www.microsoft.com/intdev/sdk/dtctrl/.

When will Visual InterDev be released, and how much does it cost?

The commercial release of the product will be available through Microsoft resellers in early spring (late March). Pricing will be announced in Q1 1997.

What platforms does Visual InterDev support?

Visual InterDev allows developers to build HTML-based Web applications that can be accessed by any browser on any platform. The development tool itself will run on Windows 95 and Windows NT Workstation 4.0.

This white paper discusses the following topics:

Web application development tools are part of a rapidly emerging software category. Because the category is so new, it is important that reviewers define a consistent set of criteria for the category and apply these criteria both to select which tools belong in the category and to fairly evaluate tools considered for review. This section assists reviewers in this process by providing information gathered from extensive customer research during the design process for Microsoft Visual InterDev.

There are four general classes of Web development tools: authoring tools, programming tools, server-side application engines, and application development tools.

- **Web authoring tools.** Web authoring tools include WYSIWYG HTML editing tools, and sometimes include additional Web site management capabilities. Examples of these tools are Sausage Software HotDog, SoftQuad HoTMetaL Pro, Adobe SiteMill and PageMill, Netscape Navigator Gold, and Microsoft FrontPage 97. While these tools often provide some support for adding applets and scripting to Web pages, their focus is on providing the content author with robust support for easy delivery of great-looking Web pages and for managing sets of pages. The target audience for these tools is the non-programmer, not the application developer. As standalone products, Web authoring tools should be reviewed in a separate category and judged by how well they meet the needs of Web content authors.
- **Programming tools.** Programming tools provide a programming language and compiler, and include tools such as Microsoft Visual J++, Microsoft Visual C++, Symantec Cafe, Sun Microsystems JavaWorkshop, and 4GL tools such as Microsoft Visual Basic, Microsoft Visual FoxPro, Borland Delphi, and Powersoft PowerBuilder. Programming tools offer critical Internet support by providing developers with the features they need for building components that may be used as part of a Web application. For example, programming tools may be used to build common gateway interface (CGI) executable files, Internet service application programming interface (ISAPI) or Netscape server application programming interface (NSAPI) applications, Java applets, ActiveX Controls, or Active Server Components. As standalone tools, however, these products are not focused on HTML-centric development. They are typically used in conjunction with other tools that provide the Web page development, dynamic HTML generation, client-side and server-side scripting, and site management — all core requirements for a Web application development tool. Programming tools can best be described as component producers, while Web application development tools such a Microsoft Visual InterDev are component consumers.
- **Server-side application engines.** Server-side Web application engines provide a framework for server-side scripting, database access, and Web state management, but they are not development tools. Rather, they provide an infrastructure upon which Web application development tools can be based. Often these application server engines are implemented as CGI-based applications or as NSAPI or ISAPI applications. While many Web development tools provide a server-side application engine of their own — Haht Software HahtSite and NetDynamic Corporation's NetDynamics Studio, for example — there are also some products available, including the current versions of NeXT WebObjects and Netscape LiveWire Pro, that are almost exclusively focused on providing the server engine as opposed to providing a visual development environment for building applications. For example, the Active Server Page feature of Microsoft Internet Information Server 3.0 provides an infrastructure for building Web applications, although it does not itself qualify as a development tool.
- **Web application development tools.** Finally, there is the category of emerging Web application development tools that are focused on delivering dynamically generated HTML pages to a Web browser. These products include Microsoft Visual InterDev, Borland Intrabuilder, Haht Software HahtSite, NetDynamic's NetDynamics Studio, and BlueStone Sapphire/Web. These products offer varying degrees of completeness. The remainder of this section offers more details about the core customer needs within this new software category, to help flesh out the core criteria these tools should be judged against. It is important to note that this emerging category of Web application development tools has important relationships with the other categories discussed above, including:
  - **Content authoring tools.** Any complete Web application development tool should include a WYSIWYG HTML editor. This ensures that programmers can easily create the portions of the HTML documents that serve as the user interface for the Web applications they develop.
  - **Programming tools.** It is critical that a Web application development tool offer developers an easy way to integrate components built using programming tools. This ensures that developers who program for the Web can easily incorporate the rich, high-performance business functionality characterized by components developed in languages such as C++, Visual Basic, Java, Pascal, COBOL, and others. Integration with components developed in these tools also ensures that a Web application can be integrated more seamlessly with existing systems already deployed within corporations.
  - **Server-side application engines.** Web application development tools must work in tight cooperation with

an infrastructure provided by a server-side application engine to fully meet developer needs. Without such a server-side infrastructure, developers who program for the Web would be unable to manage application state across distinct HTML pages, to connect Web applications to databases, or to perform the other server-side processing required to deliver functionally rich applications that fully meet user needs.

Our customer research reveals four broad areas of requirements that a Web application development tool must meet in order to be considered complete. We suggest that each of these areas be weighted evenly in order to review tools in this emerging software category.

Criterion	Percentage of Total Score
Integrated Development Environment	25%
Server-side Web Application Framework	25%
Integrated Database Tools	25%
Site Management and Content Editing Tools	25%
Total Score	**100%**

This section includes the following topics:

An integrated development environment (IDE) is a critical requirement for a Web application development tool, since it plays a fundamental role in increasing developer productivity. Research indicates that 85 percent of Web developers find the lack of a robust, integrated development environment as the primary productivity barrier when building Web applications. An IDE brings together all the tools required to build and manage a Web application within a single workspace. Specifically, a Web application development environment should include the following features:

- **Visual integrated development environment.** Developers want a visual environment so they can more rapidly build their applications by integrating the variety of tools required to build, deploy, and manage sophisticated Web applications.
- **Project system.** The development environment should support creating projects with easy access to all files that make up the Web application. A project system makes it easier for developers to track and manage their applications over time.
- **Multi-user, team development support.** Research shows that most corporate Internet and intranet sites are created by teams. So, a Web development environment must support multiple users working on the same project at the same time.
- **Integrated database development support.** Most corporate Web applications have a database component. Zona Research reports that despite the varied nature of intranet applications, a common thread is the need to have access to back-end DBMS systems. It is critical that a Web development environment offer integrated database development features so developers are not forced to switch to a different development environment when developing the database portions of their Web application.
- **High-level language development.** To be accessible by the broadest range of developers, a Web development environment should support higher-level language development (such as Visual Basic Script and/or JScript), so that lower-level programming in Java, C, or C++ is not required.
- **Support for multiple, industry-standard scripting languages.** Developers do not want to be locked in to a single scripting language for Web development. Therefore, a Web development environment should offer the option to work in standard Web scripting languages and the capability to plug in scripting languages available from third-party software vendors.
- **Support for client-side Java applets and ActiveX Controls.** A Web development environment should provide support for integrating both client-side Java applets and ActiveX Controls into Web pages. With this capability, developers are empowered to deliver rich, client-side interactivity while providing extensibility using components provided by other developers and/or other software vendors.
- **Color-coded HTML and script source editing.** Most developers want to work directly in HTML and scripting source code. They find tools that generate HTML and script but do not offer access to the generated code very limiting. So, it is critical that a Web development tool offer the flexibility to work directly with HTML and script source, including modifying generated code using a color-coded source editor.
- **WYSIWYG HTML editing.** Research shows that while most Web developers want to work directly in HTML source, they also want to create template HTML pages in WYSIWYG mode to boost their productivity. So, a Web development environment should include a state-of-the-art WYSIWYG HTML editor, integrated with the project system.
- **Seamless extensibility of IDE.** A Web development tool should provide robust extensibility architecture, so that third-party software vendors and corporate developers can seamlessly extend the tool with new capabilities.
- **Support for using third-party content editors.** Many developers already use favorite editors, such as WYSIWYG HTML and/or image editors that they will prefer to continue using. A Web application development environment should enable integration of third-party editors with the project system to offer developers with this option.
- **Integrated Help system, reference materials, and samples.** The development environment should provide excellent support for developers to get started quickly and to find answers to common questions without leaving the IDE.

Microsoft Web Developer Needs Analysis Research Project.

"Hitting the Elephant," Internet Use and Applications, Zona Research, 1995.

Microsoft Visual InterDev Design Workshops.

Because the client portion of a Web application consists of a Web browser that in many cases is only capable of reading and displaying HTML pages, much of the logic for a Web application must reside on the server computer, not on the client computer. For developers to build server-based applications, a server-side application engine is required. The three most common developer requirements that a server engine fills are:

® Database access

® State management

® Server-side scripting

While a Web development tool does not need to provide the infrastructure for these capabilities directly, it must work in conjunction with a server-side engine that does provide them. When evaluating the server-side support a Web application development tool provides, developers look for these features:

® **Open database access.** The development tool needs to provide a basic mechanism for connecting a Web site to back-end databases. Database access should be flexible, programmable, and scalable.

® **Content-centric, server-side scripting support.** A Web development tool should work in conjunction with a server-scripting engine for executing server-side scripts embedded directly in HTML pages. Using this capability, developers are not required to create and compile separate CGI applications to build server-side logic into their Web applications. As with client-side scripting, developers should be given a choice in the server scripting language they use.

® **Application state management.** A Web development environment must offer a developer the ability to easily manage state across application Web pages. For example, a developer should be able to manage variables and database connections across distinct users and applications, even as users move between different HTML pages in the application.

® **Fast execution.** Many server-side engines are implemented as CGI executable files. For faster performance and better scalability, the engine should be tightly integrated with the Web server, preferably as an in-process component, such as an ISAPI or NSAPI application.

® **Support for server-side components.** The development environment, through the application server engine, should provide a way for developers to easily invoke and integrate components developed using programming tools such as Visual Basic, Visual C++, Visual J++, MicroFocus COBOL, Borland Delphi, and PowerBuilder. Enabling component integration is a critical requirement since developers use components to integrate Web applications with existing internal systems, segment their development process, and reuse business logic across applications.

® **Support for multi-tier applications with distributed components.** In addition to enabling server-side components, the server engine should support seamless distribution of components across distinct application servers for greater scalability and fault tolerance. The distribution of components should be provided in a way that is transparent to an application developer building a Web application.

® **Scalability.** While some development tools provide a server-side engine, scalability is often sacrificed. The development tool must be based on a server-side engine that can scale to meet the needs of high-traffic Internet and intranet sites.

Most Web applications will be based on information stored in databases. A Web application development tool should not only enable basic database access, but should also provide all of the database development tools a developer needs that are integrated directly within the IDE. Specifically, developers look for:

- **Heterogeneous database connectivity.** A Web development tool must support connectivity to the most popular database systems available.
- **Database object model and programmability.** The development tool should provide an object-based, flexible programming model for manipulating database objects, transactions, and content within a Web application.
- **Visual database connection and query building.** To boost developer productivity, a Web development tool must offer developers a visual way to construct database connections and to visually build and test SQL statements without leaving the IDE.
- **Multiple database connections per page.** Developers should be able to integrate data easily from multiple databases within a single HTML page.
- **Databound HTML forms.** One of the most important Web development tasks is constructing HTML forms that are bound to databases. Traditionally, this has also been one of the hardest and most frustrating tasks for Web developers. A Web development tool should thus automate the process of building databound HTML forms.
- **Visual viewing and modification of database structures from within the IDE.** Developers should not have to leave the Web development environment to view database objects and structures such as tables, fields, stored procedures, and records; rather, features for viewing and manipulating data should be directly integrated into the IDE.
- **Visual database design and administration.** In many cases, developers must create database structures from scratch, and/or modify existing database structures while developing a Web application. The Web development tool should provide a way to design and administer server-side databases directly within the IDE. Such support should include the ability to create and alter table definitions, create and alter table keys and indexes, create stored procedures and views, and to easily create relationships between tables.

Microsoft Web Developer Needs Analysis Research Project.

Microsoft Visual InterDev Design Workshop

A Web development tool should provide support for managing complete applications throughout their life cycle. This means that a Web application development tool must provide support for application development, content editing, content publishing, and ongoing Web site management. Specifically, developers want:

- **Integrated site management.** Site management includes the ability to easily add, delete, and restructure Web content files, and to automatically detect and repair broken hyperlinks. The tool should also make it easy to set up new Web sites through tight integration with a Web server.
- **Site visualization.** Developers want to view the logical structure of their Web sites easily so they can readily determine hyperlink relationships and track and fix broken links.
- **One-button publishing and content staging.** Developers need integrated support for moving files out of local working directories to a Web server. The tool should provide support for transferring an entire Web site from a staging or development server to a live Web site server.
- **Optional source control.** The Web application development tool should offer integration with a source control system, so developers can roll back and compare changes during development. In addition, the source control integration should offer file-locking features to protect against editing conflicts in multi-user development environments.
- **HTTP project access.** Ideally, the Web project system should offer access and manipulation of Web application content using HTTP Web protocol. This functionality means that site management features of the tool can work through corporate firewalls and even through Web hosting services.
- **Content editing tools.** Besides providing for HTML source and WYSIWYG editing, the Web application development tool should provide an image editor capable of creating and manipulating images in .gif and .jpg file formats.
- **Support for WYSIWYG 2.5D layout page designs.** HTML tags are evolving to support more sophisticated layout capabilities. For developers, one of the most important new capabilities is 2.5D layout. Using 2.5D layout, developers can work with pixel-precise control over object placement and object layering. Such layout capabilities, while new to HTML, have been taken for granted in any form-based rapid application development (RAD) development system. The development environment should provide 2.5D layout capabilities based on the World Wide Web Consortium (W3C) 2.5D draft layout specification for HTML.

You can use the **Advanced** button to create a lookup field. For example, if you are creating a form based on the Students table, you can display the major description from the Majors table instead of the major ID stored in the Students table. To retrieve the major description, use the **Advanced** button to set the table to lookup and the field to display.

In Chapter 1, you took a brief look at how a Web team uses Visual InterDev as a central point to coordinate their development roles. In this topic, you will learn about where files are hosted, as a Web team progresses in their development process.

When a Web team builds a Web site, there are three different locations where files can be hosted. These locations are shown in the following illustration.

{ewc MVIMG, MVIMAGE,!W02g100.bmp}

Test Web Server

When a Web development process begins, a team member creates a project in Visual InterDev on a Web site test server. This is where the master files for a site are hosted until the site is fully developed, tested, and then published for users.

Team Development Computers

Team members edit files on their local development computers, and save them to the Web site test server. When a master file is available on the test server (no one has a working copy of the master), any team member can retrieve a copy of the master file they want to edit.

If a team is large, remotely located, or simply wants to manage file development, Microsoft Visual SourceSafe can be loaded onto the Web site test server.

Production Web Server

Once the files for the Web site are finished and have been tested, the Web team should publish the files to a secure Web server that has been removed from the development and test process. This will ensure that files are not arbitrarily changed. The team can also provide the appropriate security on the production Web server so that any malicious tampering will not occur.

Click here to connect to the Patricia Seybold Group Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The Patricia Seybold Group provides strategic guidance and tactical advice for organizations seeking business advantage through the application of information technology.

The Group offers on-line, interactive, electronic services; technology-specific publications and consulting; research reports evaluating emerging areas of technology; and customized consulting services structured to support specific organizational objectives.

Visit their Web site for more information.

Welcome to the reviewers guide for Microsoft Visual InterDev, the newest member of the Microsoft family of visual tools. This guide will help you understand the product design goals for Visual InterDev, as well as the Microsoft strategy for enabling businesses to build Web applications based on the Active Platform. The following documents are included in this guide to assist you in your evaluation:

® Microsoft Visual InterDev Reviewers Guide

® Evaluating Web Application Development Tools

® Microsoft Visual InterDev Frequently Asked Questions

® Microsoft Visual InterDev Guided Tour

Note The Microsoft Visual InterDev Guided Tour documentation is not included in this title. However, the tour can be automatically launched from Up and Running with Microsoft Visual InterDev Tutorial, in the Sample Applications section of the Library, and there is a pointer to the tutorial documentation from that topic.

As an integrated Web application development system, Visual InterDev introduces many new concepts and features. It is strongly recommended that you take the guided tour that is provided in order to fully evaluate this new development tool. You can find the guided tour files on the Visual InterDev Web site at http://www.microsoft.com/vinterdev/us/samples/. These files are also available in the Sample Applications section of the Library (as stated in the Note above). The Tutorial.exe is a self-extracting file that builds a folder on your local hard drive with the content you can use to take the tour.

Additional information is available on the Microsoft Web site, including reference materials for Microsoft Internet Information Server 3.0, Active Server Pages, Active Data Objects (ADO), ActiveX technology, Visual Basic Scripting Edition, JScript, Microsoft SQL Server 6.5, and the Microsoft FrontPage 97 Web authoring and management tool.

Click here to visit the Microsoft Workshop Home page for links to these and other areas related to Web authoring and development.
{ewc mvimg, mvimage,!intjump.bmp}

Click here to visit the Microsoft Visual InterDev Web site for the latest product-specific information.
{ewc mvimg, mvimage,!intjump.bmp}

The Reviewers Guide includes the following topics:

Over the past two years, the Internet has blossomed as the most important worldwide communications network, physically connecting consumers and organizations while also providing robust standards for publishing shared information via the World Wide Web. There is no doubt that the Internet has become the de facto *information superhighway* and Web protocols have become the information publishing standards for this highway.

Today, however, the Web is being used for more than publishing information. In fact, most commercial and corporate Internet sites can be more accurately described as Web applications because they require complex processing to create a more compelling, informative experience for users. Such Web applications can offer tremendous benefits by providing interactive functionality and seamless access to personalized information for both consumers and business users. Within corporations, Web technology is also experiencing widespread growth as an effective platform for deploying intranet applications.

In “Workgroup Computing Strategies,” published April 5, 1996, The META Group predicted that…

> “…during the next five years, there will be a major transition from existing central and client/server architectures to a user-centric information architecture, with most corporate information and applications accessible by Web clients.”

This section includes the following topics:

This section includes the following topics:

Visual InterDev is a tool specifically designed for developers who want to build sophisticated, dynamic Web applications. Visual InterDev supports team-based development and fully interoperates with the Microsoft FrontPage, a Web authoring and management tool designed for non-programmers, including end users and designers. Thus, teams of developers and nonprogrammers can work together on Web sites. As a tool for developers, Visual InterDev is based on these important design goals:

- **Integrated visual development environment.** Visual InterDev includes a comprehensive development environment that integrates all the tools necessary to build and deploy Web applications. The environment is designed to dramatically boost developer productivity.
- **Support for building Active Server Applications.** Active Server Applications are based on Active Server Pages, a new feature of Microsoft Internet Information Server 3.0. As a server-side application framework, Active Server Pages make it easy to build dynamic Web applications with sophisticated server-side processing such as database access, state management, server-side scripting, and reusable server components. Visual InterDev is the best way to build Active Server applications.
- **Powerful, integrated database tools.** Visual InterDev offers the most complete and advanced database development features available in any Web development tool. Visual InterDev provides scalable Internet- and intranet-based access to any database that supports Open Database Connectivity (ODBC). This includes high-end database management systems such as Microsoft SQL Server, Oracle, Sybase, Informix, and IBM DB/2, as well as desktop databases such as Microsoft Access and Microsoft Visual FoxPro.
- **Site management and content development tools.** Visual InterDev includes complete Web site management capabilities that are compatible with Microsoft FrontPage, including a unique site-visualization tool to aid in ongoing site-management tasks. In addition, Visual InterDev includes a version of the FrontPage WYSIWYG HTML editor as well as tools for developing Web-based images and sound effects.
- **Open and extensible.** Visual InterDev is designed to support industry standards such as HTML, HTTP, ODBC, ActiveX, COM, and Java. As such, it delivers the highest degree of interoperability with other tools and platforms and provides for seamless extensibility through third-party components.

{ewc mvimg, mvimage,!illust.bmp}
Figure 2. Microsoft Visual InterDev and Microsoft FrontPage.

Almost all Web sites are created and maintained by teams of people working together, including both programmers and nonprogrammers. The combination of Microsoft FrontPage and Microsoft Visual InterDev is an ideal workgroup solution because team members can work on the same Web site simultaneously, each using a tool tailored for the developer's specific needs. Third-party development and design tools can also be used easily in conjunction with both Visual InterDev and Microsoft FrontPage.

This section includes the following topics:

This section includes the following topics:

This section contains the following topics:

This section contains the following topics:

The open and extensible architecture of Visual InterDev is a critical aspect of the tool, since it provides a high degree of flexibility for developers. Visual InterDev delivers on the promise of openness by:

- Supporting the HTML and HTTP Internet standards.
- Supporting ODBC for connectivity to all popular database systems.
- Delivering a thin-client, cross-browser, cross-platform Web application development environment.
- Offering full client-side and server-side extensibility by way of including ActiveX Components and Java applets.
- Supporting Visual Basic Script and JScript client-side and server-side scripting as well as other scripting engines through third-party scripting engines.

Visual InterDev is an integrated, visual development tool designed to meet the needs of developers who want to build dynamic database-driven Web applications for corporate intranets and the Internet. Because Visual InterDev fully interoperates with Microsoft FrontPage — a tool designed for non-programmers and graphic designers — teams of developers and non-programmers can easily work together on Web sites. Visual InterDev delivers on the following key design goals:

® Integrated, visual development environment.

® Support for building Active Server applications.

® Powerful, integrated database tools.

® Integrated site management and content development tools.

® Openness and extensibility.

Visual InterDev is a comprehensive Web development system, including integrated programming, database development, site management, and content editing tools. Developers will find that Visual InterDev helps increase their productivity by offering a single, integrated development environment that includes easy-to-use, visual tools for building sophisticated Web applications. By using the powerful database development tools in Visual InterDev, programmers can work with any database that supports ODBC. Finally, Visual InterDev delivers HTML-based Web applications that are browser- and platform-independent while also providing open extensibility with third-party components that can be used to seamlessly extend the tool's capabilities. Corporations will find that Visual InterDev provides them with a distinct business advantage for their Internet- and intranet-based applications.

Development Tool Requirement	Description and Explanation	Microsoft Visual InterDev Support
Integrated Development Environment		
Visual Integrated Development Environment (IDE)	Includes a single, visual development environment that integrates all the tools necessary to build and deploy Web applications.	P
Project System	Supports creating projects to track and manage all elements for the Web application.	P
Multi-user, Team Development Support	Supports multiple users working on the same project at the same time with source control safety.	P
High-Level Language Development	Does not require lower-level programming such as Java, C, or C++.	P
Supports Multiple Scripting Languages	Supports Visual Basic Script and JScript as well as other plug-in scripting languages.	P
Supports Client-side Java Applets and ActiveX Controls	Integrated visual support for inserting both client-side Java applets and ActiveX Controls.	P
Color-Coded HTML Source Editing	Supports integrated, color-coded source editing for HTML with scripting code.	P
WYSIWYG HTML Editing	Includes WYSIWYG HTML editing tool, integrated with project system.	P
Seamless Extensibility of IDE	Offers robust extensibility architecture via third-party wizards and design-time controls.	P
Support for Using Third-Party Content Editors	Supports integration of third-party content editors with the project system.	P
Quick Preview of HTML Pages in Any Browser	Offers developers an easy way to preview content in multiple HTML browsers, including directly inside the IDE.	P
Integrated Help, Reference Materials, and Samples	Offers the necessary support to get developers up and running quickly.	P
Server-side Application Framework (Active Server Pages)		
Open Database Access	Offers support for connecting Web sites to most major database systems.	P
Server-side Scripting Support	Works in conjunction with Active Server Pages for executing serverside scripts.	P
Application State Management	Manages state across application pages.	P

Fast Execution	Eliminates need for out-of-process CGI executable files.	P
Support of Server-side Components Written in Any Language	For developers who want to instantiate and use server components developed in any language supporting COM through server-side scripting, including Visual Basic, C++, Java, and so forth.	P
Support for Multi-tier Applications with Distributed Components	Support for server-based components that are distributed across computers.	P
Scalability	Active Server Pages scale for high-traffic Internet and intranet applications.	P
Integrated Database Development Tools		
Heterogeneous Database Connectivity	Database tools work with multiple ODBC-based databases.	P
Database Object Model and Programmability	Provides object-based, flexible programming model for manipulating database content.	P
Visual Database Connection and Query Building	Includes visual development tools that aid in building dynamic, database-driven Web pages.	P
Multiple Database Connections per Page	Supports the ability to integrate data from multiple databases within a single HTML page.	P
Databound HTML Forms	Eases the creation of HTML forms connected to databases with the Data Form Wizard.	P
Visual Viewing and Modification of Live Database Content from IDE	Ability to view and modify database content and structure from within IDE.	P
Visual Database Design and Administration	Includes ability to create and edit SQL Server database designs, including tables, schemas, and relationships.	P
Web Site Management and Content Development		
HTTP Project Access	Offers support for managing sites over HTTP so that content can be published through firewalls.	P
Integrated Site Management	Offers integrated site management, including automatic link repair and auto-creation of new Web sites.	P
Web Site Visualization	Offers graphical representation of page relationships, including primary and secondary links, broken links, and embedded components.	P
One-Button Publishing and Content Staging	Developers can easily publish content to a Web server and move content from staging to production servers.	P

Optional Source Control	Integrates with a source control system to provide revision tracking and check-in and check-out capabilities.	P
Includes Multimedia Editing Tools	Includes feature-rich Image Composer and Music Producer editing tools.	P
Supports WYSIWYG 2.5D Layout Page Designs	Provides frame-based layout environment based on new W3C 2.5D layout specification for HTML.	P

If you have only a single home computer, you might not have thought a lot about the topic of security because you really haven't had to. Maybe you implemented some basic security measures to keep certain confidential information out of the kids' hands, like Santa's Christmas shopping list, but other than that, you probably haven't had to worry too much about protecting the data on your machine. The system is fairly closed, and therefore, no big security risk exists. The more people who have access to your system, however, the more at risk you become. If you have any organization, even a small business, in which a number of people access your server, or if you are connected to a LAN, security then becomes a significant issue. You need to think about the risks posed by all the people who have physical access to your computer. Still, while it is a real concern if someone in your organization accidentally deletes a file, say, the exposure you face is probably limited.

From a security standpoint, the Internet is a hostile environment. The number of people who could access your system and damage it either on purpose or by mistake increases dramatically. The risk of loss is even greater if you want to conduct financial transactions or if you deal with any kind of sensitive or confidential information.

Actually, you can't even install Windows NT Server without enabling some basic security measures, but there are other steps that you should take to provide the best protection you can. Our intent in this chapter is not to instill security paranoia. The odds of someone breaking into your system and wreaking havoc in one way or another are actually quite low. But nonetheless, the risks should not be ignored.

This section contains the following topics:

Human risks to your system come in many forms and can be intentional or unintentional. The potential result from any person's harmful actions is the loss of valuable information on your server.

We've all heard news stories about hackers gaining access to important information by breaking through computer security systems. Many hackers can be described as intelligent but bored people who try to break into computer systems for the fun and challenge of it. Most hacking on the Internet results in just plain and simple vandalism. Getting past a security system is like a game for these people — the challenge lies in seeing how far into a given system they can get and, for some, the more difficult the security, the more compelling the game. Once hackers crack your system, they are free to muck about with anything you have on your server. Better-mannered hackers may leave merely a signature file just to let you know they've been there. Others leave more destructive calling cards, like viruses, or in extreme, really pathological cases, they might leave you with nothing left on your server at all. The malice in this is beyond us to explain — we simply want to help you protect your site from it.

Most hacking is done just for hacking's sake, but sometimes hackers turn criminal, breaking into systems for the purpose of theft and financial gain. Hackers that cross this line are referred to as *crackers;* they are looking for specific information, like credit card and bank account numbers. For these thieves, breaking into a secure system is not a game but a serious business, so they may have an arsenal of sophisticated tools and software as a means of gaining access.

You might think you have covered all the security bases by restricting access to your computer from outside your organization. Think again. You must also consider the possibility of enemies within your system. A disgruntled employee can pose an enormous risk. Vandalism of a company's computer can be a great method of revenge for people who feel they have been mistreated or who have been fired from your organization. Less dramatic, but potentially just as devastating, are the risks of accidental deletion of information by trustworthy people or employees. Anyone who has ever worked on a computer has probably had the exhilarating experience of accidentally losing information at some point or another. People are human and mistakes happen. The bottom line is that no matter who has access to your servers, your information must be protected.

Most commonly, hackers study your system and look for any area of weakness. Common security weaknesses include holes in your system from improperly configured software that they can slip through, passwords that can be easily discovered, or information traveling across wires that can be intercepted. Your first job in implementing security is to analyze the weaknesses of your particular system. Let's consider some of the common ways that systems are penetrated.

Improperly configured software can leave holes in your security that a clever hacker can easily discover. If, for instance, you accidentally set up a user account with no password, you are essentially leaving an open door to your system. You should take the time to read the Windows NT Server documentation and really learn how security works.

In rare cases, software can contain security holes that are a result of design flaws or bugs, and hackers make it their business to keep abreast of these. When security bugs are discovered by hackers, that information is disseminated very quickly around the Internet, so it is generally a good idea to keep information about your specific server and software confidential. If a hacker knows specifically what system you are using to run your Internet presence, then the hacker can aim an informed attack at that system.

People commonly protect information and access to their computers by setting up authentication procedures using passwords. This may seem virtually foolproof, and, if used properly, it is. But no system is totally uncrackable. Hackers have methods and tools to discover passwords. For example, a hacker can launch a *dictionary attack* by using software on the hacking computer to try continually to log on to the target server using effectively each word in the dictionary as a password. It could eventually hit on the right word. Another method hackers use to discover passwords is to study any personal information available about you on the network. This often leads to the discovery of a password because many people use personal information, like children's or spouse's names or birthdates, as passwords.

Another tool used to steal information is a *packet analyzer,* also referred to as a *sniffer*. This tool, legitimately used, helps track problems in routing and performance on the Internet. Hackers, however, use packet sniffers to view the contents of packets going across the network, hoping to gain exploitable information from them. You need to be aware that whenever you send information in clear text form over the Net, you face the risk that this information could be intercepted. This is very similar to the risk you take using a wireless telephone. Anyone can be listening at any time, although it is not very likely. Just as some people give out credit card numbers over wireless telephones, you will have to calculate the risk yourself of sending sensitive information over the Internet, or you will have to consider some other way of sending the information.

One of the reasons that Internet security is such a problem is that there is no central Internet authority. This makes it fairly easy for smart individuals to fool the system into thinking they are someone they are not, an attack method known as *spoofing*. Also, it is often impossible to trace hackers on the Internet.

All this said, though, it is our opinion that the media has exaggerated the prevalence of cybercriminals and hackers. Incidents of hackers successfully breaking into computer systems are actually rare and usually occur because of improperly configured security settings.

Compared to other operating systems, Windows NT Server has one of the best security systems on the market. Microsoft proudly touts the fact that they designed security into Windows NT Server from the ground up, enabling a higher level of security than operating systems that were designed with security as an afterthought. In fact, Windows NT Server takes into account security scenarios that most people won't even think of.

This section contains the following topics:

To administer Windows NT Server security, you have to either take on the job of security administration yourself or appoint someone within your organization to be the security administrator. The security administrator carries out the security policies that you deem appropriate for your organization. The administrator needs to configure security for Windows NT Server and continually maintain security by tracking and auditing the system.

When you install Windows NT Server, a default account called *Administrator* is created that has full access to the system. You can use this account for security administration, or you can create a separate account with administrative privileges. The structure you choose for administering security depends very much on the size and nature of your organization, as does your security policy. If your organization consists only of a handful of people whom you trust and who all understand your goals for security, you might give everyone administrative privileges. If your organization is larger and you find it is necessary to keep some users out of certain areas of the system, you can have single or multiple administrators and set up other users with appropriately more restricted access. Windows NT Server gives you flexibility in the way you delegate security administration for your organization.

The Administrator account has pretty much complete access to the system, and, for this reason, it can be dangerous if someone compromises the security of this account. We recommend taking the following precautions to protect this account.

First, assign a new user name to it, something that will not make it recognizable as the administrator account. Don't use the names Admin or Root — pick something unique that other people won't think of. Because of the high degree of access the administrator account has, it is a likely target of hackers, so you might as well make it difficult for them to find the account in the first place. Second, be just as careful about choosing the password to the administrator account. Remember the basic password guidelines: don't use your name, birthday, or other personal information as your password. Servers have been broken into because the Webmaster (someone who should know better) used a password that the hacker discovered from the Webmaster's personal blurb on their Web page.

To rename the Administrator account, follow these steps:

1. Launch User Manager For Domains. (You can find it on the Administrative Tools submenu.)
2. Highlight the username Administrator, and then select the Rename option from the User menu.
3. Type in the new name when the Rename dialog box appears. (See Figure 5-1.)
4. Choose OK.

{ewc mvimg, mvimage,!illust.bmp}
Figure 5-1. How to rename the Administrator account.

If you have multiple user accounts on your system, the best way to manage their security and access to the system is to create user groups. Windows NT Server allows you to group users and apply security to each group. You might want to create groups to represent different departments in your organization or users who need access to a specific directory, but you can use any logical criteria for grouping.

As a member of a group, a user has the same security access as all other members of the group, although a single user can belong to more than one group. Access to files, directories, and other resources can be set for the entire group. To give new users access to a resource, simply make them members of the group that already has access to the resource. It is easier to keep track of the security permissions assigned to a small number of groups than to keep track of those assigned to a large number of users.

To create a group, follow these steps:

1. Launch User Manager For Domains, and select New Global Group from the User menu.

   Note Windows NT Server has two types of groups: Global and Local. Refer to the Windows NT Server documentation for more details on the differences between Global and Local groups.

   ---

2. Enter the name of the group and, if you want, a description. See Figure 5-2.

   {ewc mvimg, mvimage,!illust.bmp}
   Figure 5-2. New Global Group dialog box.

3. Select the user or users you would like to make members of the new group from the list on the right, and click the Add button.
4. Conversely, you can select specific users in the list on the left and remove them from the group by clicking the Remove button.

You don’t need to set up user accounts for basic access to public information on your Internet server. But if you have areas on your Internet server that you want to restrict access to, such as a subscription service, an employees-only section, or private membership areas, you can create single- or multiple-user accounts. You can either create one user account for this purpose where these special users would all log on using the same user account name and password, or you can give an account to each user, allowing each to log on with a unique account name and individual password. Do what makes sense to you. Keep in mind, though, that the more user accounts you create, the harder it becomes to keep track of who has access to various resources.

You must be absolutely certain when you create user accounts for access from the Internet that you give them security access for only the information that you want them to have. This probably sounds like a no-brainer, but when you allow any access to your server from the Internet, you should be extremely careful.

Individual accounts for Internet users should not be included in the Domain Users group. You should make these accounts members of some other group (perhaps Domain Guests or just Guests) and then set that new group to be the primary group to which individuals belong. After you have set a new primary group, remove the accounts from the Domain Users group. This sounds kind of mixed up, but there is good reason for that. Windows NT Server won’t allow the removal of a user from a group if that group is selected as the primary group for that user, so you have to reassign the primary group before you can remove it. By default, Windows NT Server makes the Domain Users group the primary group for all new users.

To create a new user account, follow these steps:

1. Launch User Manager For Domains, and select New User from the User menu.
2. Fill in the requested data for the new user account. See Figure 5-3.

   {ewc mvimg, mvimage,!illust.bmp}
   Figure 5-3. The New User dialog box.

3. To change group membership, click the Groups button. Make sure you don’t put the user in a group that would give him or her more security access than the user needs or you intend.
4. Remove the user from the Domain Users group. Remember, you first have to make the user a member of another group. We suggest choosing Domain Guests, which has very little security access to the system. (The Domain Users group gets some default security settings that you might not want an Internet user to have.)
5. Highlight Domain Guests (or whichever group you chose), and click the Set Primary Group button.
6. Now you can remove the user from the Domain Users group by highlighting the Domain Users group and clicking the Remove button.
7. Give the user access to any other group you want by highlighting the group name and clicking the Add button.

After you have created a new user account, you might want to set any specific file or folder access for that user. We will discuss security for files and folders later in this chapter.

We recommend that you go through user manager at least once a month to look at both individual user accounts and group memberships so that you can delete or disable accounts that are no longer needed. Keeping the number of accounts on your system down to a bare minimum will help ensure that your system is not left vulnerable to attack and will keep the server easier for you to manage. The more accounts you have on your system, the more likely it is that a hacker will be able to find a poorly secured user account.

Windows NT Server creates a default Guest account when you first install the software. Windows NT Server version 4 automatically disables that Guest account. In earlier versions, the Guest account was not disabled, which made it possible for an unknown user to be authenticated as a guest automatically. We recommend that you leave the Guest account disabled. If you want to create guestlike accounts, create a new account instead, to prevent unintentional access through the settings that are automatically given to the Guest user account.

The primary way to control access to your server is by establishing a process for identification and authentication of users on the system. When you use your bank's ATM machine, you follow an identification and authentication procedure: you insert your ATM card into the machine and then enter a secret code or a PIN (personal identification number). This identifies to the ATM who you are and what account you have access to. Identification and authentication for users of your Internet server is not much different.

Windows NT Server automatically implements identification and authentication. When you set up a user account, you are also required to choose a password that will be the user's form of authentication. You can further configure your system to set limits regarding when users can have access to the system, when they will need to change their passwords, how long they can keep a certain password, and other settings.

Windows NT Server supports basic authentication as well as Windows NT Challenge/Response. Basic authentication asks the user for a username and password and transmits that information in clear text format across the Internet. This can cause a potential security problem if someone is using monitoring software to view the packets. Windows NT Challenge/Response, on the other hand, acquires the username and password information from the operating system. The user's computer tells the server which user account the user logged onto at the workstation. The advantage to this method of authentication is that users are not required to enter their usernames and passwords when they try to access a private area. The server checks against information at the workstation to authenticate the user automatically. The drawback to this method is that it does not work with operating systems other than Windows 95 and Windows NT Workstation, and it doesn't work with browsers other than Internet Explorer 2 or later. In addition, many people might find that the user account they log onto at their workstation will be different from the user account on the Internet Information Server (IIS).

IIS allows you to choose the methods by which it will authenticate users. We recommend that you use basic authentication for areas that are private but don't contain sensitive information. Windows NT Challenge/Response should be used in environments where you know the users can log onto their workstations using the same identification as that used on your server.

This section contains the following topics:

Windows NT Server allows you to set certain parameters regarding passwords on user accounts. By configuring user and machine policies, you can set password restrictions to determine whether users will be allowed to change passwords, how often they will need to change them, and the minimum length for passwords. You can also set a feature called *account lockout* that disables the account after a given number of failed authentication attempts. It is very important to understand the options available before you make choices about the password user properties you will set.

How to Set User and Machine Policies for Security

Follow these steps to set your security policies:

1. Launch User Manager For Domains.
2. Select Account from the Policies menu.
3. In the Account Policy dialog box, set the specific Password Restrictions. See Figure 5-4.

   {ewc mvimg, mvimage,!illust.bmp}
   Figure 5-4. This is perhaps the single most important dialog box for protecting your system from would-be hackers.

Recommendations for User and Machine Policies for Security

® The Maximum Password Age is the maximum length of time a user may use the same password. We recommend that you allow users to keep a password for no more than 60 days.

® The Minimum Password Age refers to the minimum length of time that a user must have a new password before being allowed to change it again. Normally, you can set this to allow changes immediately.

® The Minimum Password Length refers to the minimum numbers of characters that a password can have. We strongly recommend against allowing passwords to be blank. A safe minimum length would be six characters, but the more the better.

® The Password Uniqueness option can be used to keep people from reusing the same password. This option forces them to choose a password they've never used before for a specified number of times. When a user has changed her password this many times, she can repeat the sequence of former passwords if she chooses. Our recommendation is that you put the password uniqueness setting to at least eight.

® Account Lockout is a very important feature, and we recommend that you enable it. This feature monitors the number of invalid logon attempts and locks the account if the number of attempts exceeds the parameters you chose. You must allow for the possibility that from time to time a user forgets a password or types it incorrectly, so we recommend that you set the number of bad logon attempts before lockout to four. You must also choose a length of time to reset the count. We recommend thirty minutes for this option. The lockout duration is the amount of time an account that is locked out remains locked out. You should set this to be at least one hour.

® The final security option is a check box allowing you to determine whether users must log on in order to change their passwords. We highly recommend that you select this option.

Anyone who has access to your computer should be made aware of some basic rules for creating passwords. The following is a list of guidelines to follow in choosing passwords.

® As a general rule, don't allow password changes over the Internet; they could be intercepted.

® Never allow passwords to be blank. This defeats the whole purpose of authentication.

® Never use as your password your name, your spouse's name, the names of your children, your birthday, anniversary, or any other information about yourself that could be publicly available.

® To foil the hacker who uses the dictionary approach, create passwords that are not real words. Mixing numerals and punctuation marks with letters makes passwords even more difficult to crack.

® The longer the password the better. We recommend that you use at least six characters.

® Change your password frequently, at least every sixty days, even if the security settings for your system don't force you to change it that often.

® Never write your password down, and never give it out to anyone. If you absolutely can't remember your password without writing it down, for heaven's sake, don't label it "password" and don't store it anywhere near your computer.

® Never send passwords across the Internet. Someone could always be "listening."

It's a fine line between making user accounts and passwords that no one can remember and making them secure enough that your system is protected, but with common sense (and trial and error), you can protect your system without making it too hard to use.

Similar rules should apply in choosing user account names as in choosing passwords. Someone trying to break into your system by getting past your authentication will have to choose an account to target, so you don't want your account names to be obvious. For instance, if a hacker knows that a "Bill Smith" has an account on your server, he may start his attack with the username *BSmith* or *BillS*. If this is the formula you actually use to create names for your user accounts, then a hacker will be one step closer to getting into your system. You may want to give this employee a username like *BillS1*. This might seem like security overkill, but it is an easy way of making your system a little more secure.

When you install Windows NT Server, certain default security settings are activated that might not be appropriate for someone with a presence on the Internet. One setting that you will need to change is the ability for users to do remote management of your server. Having remote access allows a user to monitor certain functions of the server over the Internet. In the wrong hands, this tool could provide information about the various services running on the server, the processor power and the kind of processor, the amount of memory on the machine, and pretty much all the details about the type of machine and the services it is running. A smart hacker, knowing this much about your system, could use known holes in your software to gain access or could know exactly what it would take to overload your server.

Here's how to remove remote management capabilities:

1. Launch User Manager For Domains.
2. Select User Rights from the Policies menu. See Figure 5-5 on the following page.

   {ewc mvimg, mvimage,!illust.bmp}
   Figure 5-5. The User Rights Policy dialog box.

3. Select the Show Advanced User Rights option at the bottom of the dialog box.
4. Using the drop-down list of user rights, select each right, one by one, and make sure that the users who have been given this right do not include Everyone, Guests, or Domain Guests.
5. When you have gone through all the user rights, click the OK button.

Note Pay special attention to the rights about the shutdown commands and accessing the computer from the network. These are very important user rights that very few users should have.

This section contains the following topics:

Firewalls apply only to users who have a network of computers rather than just a single server to protect. If you have a network hooked up to the Internet, a firewall is a device that you could implement in order to protect your network from unauthorized access. It acts as a semipermeable barrier that allows only packets of information meeting specific criteria to pass through. The firewall sits in between your network and the Internet and filters traffic to your network. The firewall evaluates all packets coming in and checks them against a list of packet types and source and destination information to determine whether the packet is safe. This effectively blocks potential hackers from accessing anything on the other side of the firewall. The criteria for access are configured by the administrator. If you have an Internet server, it needs to be on the outside of the firewall.

This solution is still not foolproof. There are ways to hack into a firewall. Since firewalls are based on authentication, if someone can crack a password he can get past a firewall. For ultimate foolproof protection, you should not load the TCP/IP protocol on the non-Internet servers on your network, especially if your servers contain sensitive information. That way, if someone actually manages to get past the firewall and into your network, the invader won't be able to access anything on your servers because the machines won't be speaking the same language. See Figure 5-6.

{ewc mvimg, mvimage,!illust.bmp}
Figure 5-6. A firewall between an Internet server and a network.

Windows NT Server has a logging system that keeps track of all events that happen on your system, including hardware problems, driver problems, network problems, and others. There is also an audit log that, when activated, keeps track of who accessed what files and when. Although the auditing system may sound like a very useful feature, it has a considerable drawback in that it adversely affects the performance of your machine. If you activate auditing, it will slow your system down by approximately 30 percent, so we do not recommend using it.

As an alternative to auditing, the IIS has Web and FTP logs that should provide sufficient information for your needs. The IIS log tracks the IP addresses of all users who access your server, whether they logged on with a user account, and which pages they accessed.

Windows NT security allows you to set access permissions on files and directories to protect them from unauthorized use by specific users or user groups. When you are authenticated on the server, the procedure creates a *token* that is used to identify you for security purposes throughout the system. The token carries with it a unique identification number for you and a list of all the groups of which you are a member. When you attempt to access a file, the token is checked against the permissions that are set on that file to see whether you or any of the groups that you belong to have access to the file and the level of access that you have. If your token matches the permissions, you will get access to the file.

This section contains the following topics:

The next few sets of procedures tell you how to set, deny, or change user access to files and folders.

1. Double-click the My Computer icon on your desktop, and double-click the icon of the hard disk to which you want to apply security. See Figure 5-7.
2. Search through the folder icons until you get to the folder or file on which you wish to set security, and highlight that file or folder.

   {ewc mvimg, mvimage,!illust.bmp}
   Figure 5-7. Using Windows Explorer to find a file you want to set permissions for.

3. Click the right mouse button once to open the context menu for the file or folder, and select Properties from this menu. See Figure 5-8.

   {ewc mvimg, mvimage,!illust.bmp}
   Figure 5-8. The context menu on a directory.

4. Choose the Security tab, and click the Permissions button.
5. If you selected a file, you will see a dialog box similar to the one in Figure 5-9. If you selected a folder, you will see a dialog box similar to the one in Figure 5-10. Both of these dialog boxes display the current security settings for the file or folder.

   {ewc mvimg, mvimage,!illust.bmp}
   Figure 5-9. Security permissions for a file.

   {ewc mvimg, mvimage,!illust.bmp}
   Figure 5-10. Security permissions for a directory.

If you want to remove access for a user or a group for this file or folder, then highlight the username in the list shown. Click the Remove button.

If you want to change a user or a group's permission, highlight the username or group name and choose from the drop-down list at the bottom of the dialog box to set the new security for this user or group.

If you want to add a user or a group to those who have access to a file or folder, follow these steps:

1. Click the Add button. This will display the Add Users And Groups dialog box. See Figure 5-11.

   {ewc mvimg, mvimage,!illust.bmp}
   Figure 5-11. The Add Users And Groups dialog box.

2. Then select the user or group to whom you want to grant access by highlighting the group or username. Click the Add button.

   Note In order to see individual users, you must click the Show Users button. Otherwise you will see only groups listed here.

   ---

3. Go to the drop-down list box at the bottom of the dialog box, and choose the security level you wish to provide. This security setting will apply to all the users and groups you selected to add.

If you are changing permissions for a file, that is all there is to it, and you can click OK. But if you are setting permissions for a folder, you need to decide about two more options:

® The first option is Replace Permissions On Subdirectories. Choosing this option depends on whether you want this security change to affect all subdirectories that are inside of the subdirectory or folder you are working on now. In most cases, you will choose this option, but you will need to decide that for your individual case.

® The second option is Replace Permissions On Existing Files. We recommend that you leave this option checked so that your change affects any other files that already exist in your subdirectory. That way, old and new files will have the same security setting.

You should be careful about how you set file security so as not to lock yourself out of your own drive by removing all permissions on the file or folder in question. If you do this, it is very difficult to get back in. You will have to reinstall Windows NT Server and reset security on the drive by using the security editor. Of course, if you made backups of your data, you can always recover from this kind of accident.

Note Protect your non-Internet files from the IIS account. When you install IIS, it creates a user account that IIS uses to gain access to your server. Access to files from this account constitute what will be your publicly accessible Web pages. Any areas of either your private Internet site or content that you don't want on your Internet site at all should be protected from this user account.

Before you set specific security for files, you should change the security permissions for the root directories of your NTFS drives. By default, Windows NT assigns the group Everyone with Full Control in the root. For a workstation, especially if it's not connected to a network, this setting would likely be acceptable, but for an Internet server or any other network machine, you need to change this setting. Specifically, you need to remove Full Control from Everyone at the root directory. When you do this, make sure you select the option to Replace Permissions On Subdirectories. This ensures that the security you set at the root will apply to the entire drive.

If you set permissions first on directory files and then set permissions on your directories starting at the root, your file permission settings would be overwritten. For this reason, we recommend that you set the standard permissions on directories first, and then set permissions for individual files if you need to.

Note Files or folders on Windows NT Server inherit the security permissions from their parent directory. If you have a folder with a certain level of security, any file that you have in that folder will take on the same permission.

When you set up IIS, you were asked to pick a directory or folder to act as your root folder. We suggested that you call it WWWROOT. This is the home or root directory for your Internet site. In order for the Web files to be public, IUSR_*machinename* must have access to the root. (This is the user account the system uses to access data when it doesn't have details about the user.) When you installed IIS, you were asked where to place your WWWROOT folder; this is the default root for your Web server. You were also asked where to place Gopher and FTP files, if you selected those services. The setup process by default created the IUSR_*machinename* account and gave it permissions to these default folders. As you create your subdirectories under these root directories and as you add virtual directories to your root, you want to make sure that all areas are also accessible by the IUSR_*machinename* user.

To set up files on your Internet server, we recommend that you design and implement a directory tree structure. See Figure 5-12 on the following page. For example, if you have multiple Web sites, you might have a different directory for each one. Inside each directory would be subdirectories for logical categories like images, HTML documents, and forms.

{ewc mvimg, mvimage,!illust.bmp}
Figure 5-12. A possible directory tree structure for a Web server.

To set security permissions for your Internet site, you should go through the directory structure and make sure that all directories that are going to contain public data intended for all Internet users are readable by the IUSR_*machinename* user. For files that you wish to make publicly available, you should make sure that the IUSR_*machinename* user account has at least Read permissions in the areas of the WWWROOT directory tree. If this user account doesn't have Read access to a directory, the IIS treats that directory as a private area and requires additional authentication from the user.

We do not recommend that you give the IUSR_*machinename* account access higher than Read in areas where you would not want the end user to possibly erase, change, or delete files. The only place your users might need to have Write access is in an FTP upload directory or for some kind of form that needs to be written to the disk after the user has filled it out.

Note If you do use forms, we recommend that you send them to a database like SQL Server or Access.

Sometimes you might need to create private areas on your Internet site that only certain people can access. This might be for a special pay-for-use area, a members-only area, or some private/sensitive data such as financial or medical information that should be accessible only to the correct user. In order to set this up, you would first verify that the IUSR_*machinename* account has no access to the private data. Then you would create user accounts for the people whom you want to have access to the private areas and give them appropriate permissions.

After you set up your private user account or accounts, you need to provide access to the directories for those user accounts. We recommend that you create a user group to represent each secure file area and either make user accounts members or remove them from the membership in order to control access. That way, you need to set file security only once for the group, and you can manage the membership list of the group rather than the many levels of file security that could come about as time goes on.

An IPP might want to set up private areas to allow users to update their own personal Web spaces themselves. You can create this ability by setting up special directories that the FTP server can access and creating a user account for each Web customer. You would give them Change access to the directory that corresponds to their private Web space. Then create a virtual directory for the Web server and include that private folder in the Web directory space. In this way, Web customers can use FTP to update and add to their own private areas and not be able to change any other areas.

A drop box gives users the ability to send you files without allowing others to view that information. A drop box is like a mailbox: you can put things in, but you can't get them back out, and you can't see the contents of the mailbox. Drop boxes are useful places to privately hand off information from one user to another. For example, a user could send you a new program, but, until you verify that it is virus-free, you don't want to make it available to download. Drop boxes are also useful for an FTP site, where you might want to allow users to send you files without allowing those users to see the contents of the directory where they drop off the files.

In order to set up a drop box, follow these steps:

1. First make sure the directory is in the virtual or physical root for IIS. You can do this by using the Internet Server Manager.
2. In Windows Explorer, right-click the folder that represents the drop box and choose Properties. Select the Security tab.
3. Click the Permissions button. Then remove access to this folder for everyone except administrators.
4. Add the Internet Information Server user account, which is typically called IUSR_*machinename*. Give it Read access.
5. Highlight the username, and select Special Directory Access. Uncheck Read and Execute, and add Write.
6. Select Special File Access, uncheck Read and Execute, and select Write. Make sure that Replace Permissions On Subdirectories is selected, and then click OK.

Following are the security settings that you can apply to files and directories with explanations of the amount of access that they give. The first seven permissions listed here should meet the security needs for nearly all cases. The last two that we mention are for the advanced user or security professionals.

NO ACCESS

This setting removes all access to the selected file or folder for the specified user or group. Normally you would not need to set this permission in order to keep users out of a directory. A user would not have access to a specific resource unless security was configured incorrectly. This setting removes any access to the file/folder that the user might have been granted by virtue of access to the parent directory.

LIST

This setting applies to a folder only. When applied, List permission allows the user or group to see the names of all the files and folders that are inside the specified folder. If List permission is turned off, users will not be able to see the contents of the folder. If, however, users know the name of a file within the folder file, they would not be prevented from accessing this file unless another permission has been set to prevent them from seeing it.

READ

This setting gives the user or group permission to access the contents of the folder or the files in the folder. If Read permission is not set, the resource is not accessible for viewing.

ADD

Add permission can be applied to a folder only, and it gives the user or group the ability to create new files and folders inside the specific folder.

ADD & READ

Add & Read permission applies to folders only. This setting is just a quicker way to apply the Add permission and the Read permission to a user or group. There is no difference between giving a user add and read rights using this option or using the individual permissions.

CHANGE

This security setting allows the user or group to modify an existing file or the files in a folder. With this permission, a user can overwrite or erase the contents of the file. Extreme care should be given to consideration of who should get this permission, since, in the wrong hands, Change permission can result in loss of data.

FULL CONTROL

This security setting gives the user or group all access rights that are available for the file or folder. Extreme care should be given for this security permission. Only administrators should be given this kind of access. A user with this level of permission can not only change or erase the file or folder in question but also change the security on it and lock out legitimate users.

SPECIAL DIRECTORY ACCESS

This security setting applies to folders only. It allows you to get to a lower level of security than you could get to with the previously mentioned security settings and set more specific rights. We recommend you do not use this option, unless you are experienced with Windows NT Server security, since these settings are easy to interpret. For more details, see the Windows NT Server documentation.

SPECIAL FILE ACCESS

Special File Access is similar to Special Directory Access but, as indicated in the title, applies to files rather than directories. It allows you to set lower level permissions on file access. Please consult the Windows NT Server documentation before applying any of this option’s security permissions.

Scripts extend the functionality of your Internet server. Basically, scripts are small programs that you can use to do common tasks such as send data entered on an HTML form to a SQL Server database, or send the latest stock prices to a user. Some common types of scripts are CGI (Common Gateway Interface) programs, ISAPI (Internet Server API) programs, and PERL scripts. Scripting is a very powerful ability, but there are some risks associated with running them. Since a script is actually a process or program running on your Internet server, it is possible that misbehaving code could cause problems. On top of that, if you support a scripting language like PERL, then it is possible that hackers could send their own rogue commands instead of your intended commands to do nasty things to your server, such as erase the hard disk.

Windows NT Server and IIS have ways of limiting the exposure to the risks posed by scripts. There is a special permission you must set for a directory in order for IIS to actually execute commands or scripts from it. Without this setting, IIS won't launch these programs. Activating this setting is especially important if you allow users to update their own Web pages on a server. Also, it prevents someone from typing in a command and getting something to launch that shouldn't.

We strongly recommend against using PERL or other free-form scripting languages without serious consideration of the security risks. If you are not well versed in these scripting languages, don't use them on your Internet server.

When you specify directories and virtual directories, you can enable certain IIS security settings for the Web server. These are Read, Execute, and Require SSL. The Read option allows the contents of a specified directory to be transmitted over the Internet and viewed. The Execute option allows scripts and other programs in the directory to be executed by users of the Web server. The Execute setting should be assigned only to directories that contain the scripts you approve and should not be assigned to all directories. The SSL (Secure Sockets Layer) option allows you to specify encryption for a given resource when it is accessed over the Internet.

Unless you have acquired an SSL certificate, you won't be able to select this option. It is highly recommended that you do not give Read and Execute permissions to the same directory. A directory that contains scripts doesn't need Read, and a directory equipped with a Read permission shouldn't contain scripts. This applies an extra level of security to prevent hackers from discovering how your scripts work and trying to use that to help break into your system.

Whenever you send information over the Internet, or any network for that matter, there is a risk that the information will be intercepted. For most people hosting basic Internet sites, it hardly matters if more people see your stuff. If, however, you plan to perform electronic transactions or transmit any kind of sensitive matter over the Internet, be aware that someone might be monitoring and intercepting the data. For this reason, you should consider either implementing encryption to protect information as it is transmitted or using alternative methods for exchanging sensitive data, like the use of a financial clearing house.

This section contains the following topics:

People dealing with highly sensitive information often employ encryption to scramble the data so that it is unreadable unless you have a key to it. This does not prevent the packets from being intercepted and viewed, but it makes them unreadable and therefore useless.

Information is encrypted with the use of complicated mathematical algorithms. When the data is received by the intended recipient, it is reassembled in its original form using the key that deciphers the message. Encryption requires client and server software and requires both parties to have an agreed-upon method of coding the information.

When you send encrypted data, you need to be able to prove your identity to the receiver. One way to do this is to purchase a certificate from a trusted third party. The certificate provides a method to uniquely identify the sender. This is comparable to the certificate received when you purchase something like a rare piece of art. If you resell the art, you use the certificate to prove authenticity. The concept of a certificate has been implemented in the computer world to act as a kind of digital signature.

Certificates are usually assigned by an organization that is known and trusted as a certificate source. Certifiers determine that you are who you say you are and then assign a certificate in the form of a unique set of numbers that are known only to you. These numbers are used in the various encryption formulas to encrypt and decrypt your information and are known as keys. You use a private key to encrypt the data, and a public key is given to the people you communicate with and is used to decipher the information. Certificates provide a highly secure method of identifying the source of the information and protecting the data as it travels over the Internet.

SSL is a protocol used to encrypt data that is transmitted across the Internet. SSL runs on Internet Information Server. If you enable SSL, you can have a very secure site.

In order to use SSL, the first thing you must do is acquire a certificate. Once you have it, you can enter it into the IIS software by running SETKEY. SSL security slows down the server data transfer rates, so SSL is not usually set for an entire server but rather on specific directories. When you add a new virtual directory to the server, you can enable SSL by selecting the Requires SSL check box. You must make sure that you refer to any links from your nonencrypted data to the SSL-protected area with *https*:// rather than *http*://.

How to get an SSL Certificate

Contact VeriSign to get a certificate. VeriSign is currently the only company that issues certificates for SSL. You will be required to send them some documentation to identify yourself, and you will need to purchase the certificate. At this time, the annual fee is $295.00. Refer to the URL *http://www.verisign.com/microsoft* in order to get the latest instructions for acquiring certificates.

PCT is an enhanced protocol very similar to SSL. The main advantage of PCT over SSL is that PCT encrypts your user authentication information with a different and more complex algorithm than is used for the data. This helps ensure that not only is the data protected, but the authentication information, which *must* be kept secret, is protected even more. You might wonder why you don't just apply the more secure encryption to all the data you send. Keep in mind that encryption requires a significant amount of processing power, so the stronger the encryption, the slower the system will become.

SET isn’t so much a security protocol as it is a secure method of handling financial transactions. Some people are uneasy buying goods over the Internet, either because they are uncomfortable sending their credit card numbers and addresses over the Net or because they are unfamiliar with the vendors. When you use SET to purchase something from a vendor, SET sends a packet of information to your credit card company that includes information about the amount of the transaction and the vendor number. The vendor then receives a confirmation from the financial institution that the transaction has taken place and that the bank has a voucher for the money. Using SET, you never have to give your credit card to the merchant. While IIS by itself doesn’t support SET, a new product from Microsoft, Merchant Server, will handle this.

A virus is an undesirable, self-replicating program that gets into your computer system, usually through a floppy disk or through a network. Viruses originate at the hands of people who for some twisted reason think infecting other people's computers is fun. Viruses have a trigger, meaning that when a certain condition is met, the virus activates. They are also designed to spread from one computer to another. Some viruses can be incredibly destructive to your system. They can destroy all your data or do anything else the writer can dream up. Over a network, viruses can spread like chicken pox through a kindergarten.

Your best measure to take against viruses is prevention. This means you must control access to your system. If you have only a home computer and you are careful about the source of the software that goes onto your machine, you probably will not have to worry about viruses getting on your machine. Just make sure that you don't use a floppy disk that has been *anywhere* else. With virus technology becoming more sophisticated and the potential for great damage to sensitive information, some large companies don't permit employees to bring in disks from the outside at all.

Another good measure of prevention, besides controlling access to your computer, is to purchase antivirus software and run it regularly. Of course, you must remember that antivirus software becomes dated. New strains of viruses are being introduced all the time, and the antivirus software you have might not detect a new virus on your system. Besides running the antivirus software regularly, you will have to make sure your antivirus software is up-to-date.

If your computer is connected to a network or to the Internet, your risk factor for contracting a virus increases. One good rule to follow is to never download software from a network or the Internet directly onto your hard disk. Always download new software onto a floppy, and then run an antivirus program before installing it onto your hard drive.

The last measure of prevention is, of course, common sense. Back up your system religiously, not just in case of viruses, but for any of the many problems that can happen.

No matter how many measures you take to secure your data with software, this will not protect you from someone who walks into your office with a sledgehammer and lets loose on your computer. Okay, so this isn't a very likely scenario, but you should give some consideration to the physical security of your Internet server. This is really the most basic security for your machine and is based purely on common sense. Like any other important and expensive piece of equipment, your computer is at risk from theft, vandalism, and otherwise prying fingers (like those of curious two-year-olds). If you are concerned with the physical security of your server, you will want to keep it safe from direct attack. The best way to physically protect your server is to keep it in a locked computer room or closet that only you and your trusted employees have access to. You might also want to consider putting a lock on your computer. Most come with locks, but they are about as secure as the locks on your child's diary or your suitcase. You can buy sturdier locks for your computer. Of course, they will not protect you from serious vandals or thieves, but they will keep most people from gaining direct access to the computer. As a theft deterrent, you might consider bolting your computer down to a counter or desk.

On an intranet site, security is both more of a concern and less of a concern: less of a concern because you are not opening up your system to the Internet, and more of a concern because of potential threats from within your organization and because of the kind of information you might have on your intranet. For the most part, security concerns for an intranet site are identical to those of an Internet site. You just need to consider your audience a little differently and make plans for security accordingly. Some of the main considerations for security on an intranet site follow.

As we mentioned earlier in this chapter, a disgruntled employee who has access to your network can do a considerable amount of damage. One could potentially hack into areas of the intranet that are restricted, such as files that contain your financial data. Most employees have a level of integrity that makes this risk small, but if your intranet includes information of a very confidential nature, the risk factor escalates. Someone could also simply make a mistake by setting improper permissions on a file, thereby making sensitive information available companywide.

Consider using encryption to protect some of the information on your intranet. It may seem strange or unnecessary to think about encrypting information traveling across the network in your own organization, but review the kind of information that could be found on your intranet. As your organization's intranet grows and begins to replace legacy systems for some services such as benefits administration and financial services, you should protect that data as it moves from one computer to another.

If your organization has both an Internet site and an intranet, you will have to be conscious of keeping these two networks separate. Intranet sites often contain content that would not be suitable for public consumption, so make sure that your intranet is not exposed to the Internet. This is even more critical because Internets and intranets use the same protocols to communicate. Having a firewall in place between the Internet connection and your intranet server is mandatory.

Oh, by the way, did we mention backup? Despite all the security precautions that you can take, you still can somehow lose all your information, be it by a hacker, a system failure, a curious two-year-old, or a spilled cup of java. Backup can be an effective security precaution because in many cases when a hacker breaks into your system, the goal is to wipe your system clean. But total losses can also come of accidental disruption to the server by power loss, pulled cables, spillage, or other unnatural disasters. By having regular backups, recovery from such an attack is usually quick and fairly painless. In Chapter 3, we recommended tape backup for those of you who are really concerned with potential loss of information. It is easy to use and to install, not very expensive, and it is supported by Windows NT Server. Of course, any backup system is absolutely worthless to you if you do not use it. It's important that you get into a regular routine of backing up your server, at least once every week, if not every day.

® Rename the Administrator account (highly recommended).

® Set strong password settings for all user accounts (highly recommended).

® Remove the security access for the Everyone group from hard disks (highly recommended).

® Set group security according to your particular needs (moderately recommended).

® Make sure the user IUSR_*machinename* has at least Read access to all files you want publicly accessible (highly recommended).

® For private areas of your Internet server, make sure IUSR_*machinename* has no permissions and that the user or group that can access it has at least Read access (only if needed).

® For secure encrypted sites, get and install a certificate for SSL (only if needed).

® Verify that IUSR_*machinename* and any other Internet users have no access to files that you do not want published on the Internet (highly recommended).

® Verify that only the scripts directory has Execute permission and that there are no scripts that could be used against you (highly recommended).

® Verify the physical security of your Internet server, and make sure that it is in a place safe from damage, theft, and vandalism (highly recommended).

® Back up regularly — at least weekly (highly recommended).

When setting the CODEBASE attribute, you can specify a version number only as shown in this following example code:

```
<OBJECT
    CLASSID="CLSID:7823A620-9DD9-11CF-A662-00AA00C066D2"
    CODEBASE="#Version=4,70,0,1165">
```

If you do not specify a location in the CODEBASE attribute, but you do specify a version number, Internet Component Download will search the Internet Search Path for the correct version of the control to download.

But as the Web continues to evolve from a document publishing platform to a platform for dynamic applications, a variety of development issues arise.

On the client computer, the Web pages themselves increasingly contain programming logic such as JScript or Visual Basic Script, as well as embedded software components such as Java applets and ActiveX Controls that can provide advanced functionality to users. You can review these client-side elements of a Web application in Figure 1.

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. Typical Web application showing client and server elements.

A Web application can consist of many components. Box 1 in Figure 1 depicts application logic that runs on the client computer. On the client computer, a Web application can include HTML pages with client-side software components such as ActiveX Controls and Java applets running inside the Web browser. The client pages can also include scripting that runs inside the browser, such as Web applications created with Visual Basic Script and/or JScript. Web applications can also require complex server-side processing. Boxes 2, 3, and 4 depict common server-side components for a Web application. The Web server computer is responsible for serving up HTML pages, including dispatching requests to external applications responsible for constructing dynamic HTML pages on the fly. For example, Web applications typically call external server applications to process HTML forms into a database server and to retrieve database records and format them into dynamic HTML pages, which the server application then sends to the client application for display.

On the server, dynamic Web applications must coordinate a variety of components and processing to deliver user interactivity and up-to-date information, such as real-time data and dynamic information stored in databases. These Web applications have server-side processing, most often accomplished with the use of common gateway interface (CGI) applications, to process forms, respond to user input, and format database information into dynamically constructed HTML pages.

Often the applications must integrate with an organization's existing systems, such as product and customer databases, as well as order processing and various transaction-oriented systems. (Review shaded boxes 2-4 in Figure 1.)

Just as Internet Web sites are evolving into sophisticated Web applications, corporations are also beginning to use Web technologies within their private corporate networks, or intranets, as an effective way to build and deploy internal applications. For information systems (IS) organizations, intranets provide the following core benefits for internal application development:

- **Decreased deployment costs.** Because intranet applications are server-based, the IS department does not have to distribute client-side software or reconfigure users' desktops. Rather, users can navigate directly to the intranet site with seamless access to the application and no additional setup or desktop configuration. If the application functionality needs to be changed, IS professionals can make updates to server-based code, and all users will instantly be upgraded with the new capabilities the next time they navigate to the application. In organizations with thousands of desktops potentially distributed throughout hundreds of remote offices, this benefit alone can justify the use of Web technology for internal application deployment.
- **Cross-platform applications.** Intranet applications are delivered as HTML pages and are cross-platform by nature. Organizations with heterogeneous desktop platforms are assured that all users can access the application without leaving some users out or requiring different code bases for different platforms.
- **Low-bandwidth applications.** With Web applications, most processing is done on the server and only HTML pages are delivered to the client computer, sometimes with small, embedded software components and scripting information. Thus, Web applications are automatically suited for low-bandwidth connections. This is convenient for organizations with mobile workers, such as sales agents who need to dial in from the field to access corporate information, or workers who dial in to work from their homes.

Although these benefits are tangible, IS organizations are constrained because the tools to support Web application development are immature at best — often consisting of little more than a simple text editor.

Tools for developing Web applications are scarce because the Web introduces new technologies that traditional client/server tools are not designed to handle. Specifically, the underlying network and user interface technologies for Web applications are fundamentally different from traditional client/server applications. For example, traditional client/server development environments can rely on persistent connections between the clients and servers. The Web network protocol is HTTP, and connections are intermittent — constantly being established, broken, and re-established as the browser requests pages and objects over the network. So while traditional client-server applications can easily maintain state between their various windows — for example, using globally scoped variables — Web applications are a series of loosely connected pages with no notion of persistent state between them.

Also, user interfaces for traditional client/server development using 3GL and 4GL tools are typically based on tool-specific forms technologies. These tools leverage the underlying operating system to present the various elements of the user interface. A Web-based graphical user interface, on the other hand, is based on a page-browsing metaphor and is constructed using HTML — a document-publishing standard that works across platforms.

Because of these fundamental differences, a new category of development tools geared toward Web technologies needs to evolve. To meet this challenge, Microsoft is introducing Microsoft Visual InterDev, an integrated development system for building dynamic Web applications for corporate intranets and the Internet.

Besides sharing the familiar appearance of other Microsoft visual development tools, Visual InterDev makes it easy to integrate components and business processes developed using Microsoft Visual Basic programming system, Visual C++ development system, Visual J++ development software, and Visual FoxPro database management system. It is just as easy to integrate a variety of third-party development tools within the context of an overall Internet or intranet solution, thus leveraging an organization's investment in existing tools. In short, Visual InterDev makes it easy to integrate client/server development with the Web technologies to develop Web applications that can provide a distinct business advantage.

Visual InterDev includes a variety of visual tools within a single, well-integrated development environment. A developer can create and manage shared Web projects, and Visual InterDev will automatically publish the content developed to a Web server and provide ongoing site management capabilities for that Web site. The Integrated Development Environment (IDE) also includes visual database tools for creating and managing data-driven Web sites where Web pages are dynamically constructed based on live connections to databases.

The Visual InterDev IDE is based on the next-generation Microsoft Developer Studio shell, first introduced with Microsoft Visual C++ and also used by Microsoft Visual J++. Developers who use these tools will be immediately familiar with the IDE, and new users will find it intuitive and easy to learn. It is important to note that the IDE provides an integrated, global workspace for Web sites created with Visual InterDev and projects created with Visual C++ and Visual J++. A developer can host multiple projects of these types simultaneously within the global workspace. For example, a developer might use a workspace hosting both a Visual J++ Java applet project and Visual InterDev Web project so that the developer can develop and test the applet within the same Web site where it will be deployed.

A project created with Visual InterDev consists of a live Web site. Unlike the process involving traditional client/server tools, when a developer opens a project, a live view of a site as it exists on the Web server is opened. The server can be a developer's personal development server running on a local workstation, but more typically, it will be a staging or production Web server running on a network.

The IDE is thus a complete Web site management tool the developer can use to easily modify the structure of a Web site and to edit, add, move, rename, and delete files and folders on the site. Multiple Web sites, or projects, can be open at the same time. The Visual InterDev IDE includes a main File View that a developer can use to view and navigate a Web site using the Windows Explorer-based application. The Visual InterDev IDE presents the entire Web site, including all content, such as HTML pages, GIF/JPG images, controls and applets, and other files, as well as the complete subdirectory structure of the site.

To edit or examine an element of the site, the developer clicks the file. It is then copied from the Web server to the developer's workstation and opened in the appropriate editor for that file type. Once the changes are made, the developer can save the changes locally or choose to save the changes automatically to the Web server.

The project model is multi-user: Multiple developers can work simultaneously on the same Web site. Check-in and check-out capabilities are provided by integration with the Microsoft Visual SourceSafe version control system. The project system makes use of the server-side extensions in FrontPage so that developers using Visual InterDev and users of FrontPage can work together on the same Web sites at the same time.

The IDE makes it easy for a developer to create new sites with wizard technology and to import content, including entire directory trees, into an existing site. Developers can also browse pages in the project directly inside the IDE, an important productivity feature, since a developer has no need to pull up a separate Web browser window to view a site under construction. The integrated browser is the Microsoft Internet Explorer 3.0 Web browser running as a component within the IDE. Thus, it supports Java applets, ActiveX Controls, Active Documents, scripting, style sheets, and advanced HTML features such as tables and frames.

The IDE also provides a preview-in-browser feature so a developer can set up other browsers, such as Microsoft Internet Explorer 3.0 and Netscape Navigator. Then with a single mouse click, a developer can preview any page in separate windows running those browsers.

{ewc mvimg, mvimage,!illust.bmp}
Figure 3. Microsoft InterDev Integrated Development Environment (IDE).

Microsoft Visual InterDev is an IDE for building dynamic Web applications for corporate intranets and the Internet. The IDE brings together a variety of visual tools that help increase the productivity of Web developers building sophisticated Web sites. Visual InterDev shares the same IDE as Microsoft Visual J++ and Microsoft Visual C++, enabling developers to have projects using Visual C++, Visual J++, and Visual InterDev open simultaneously within a single, global work space.

Design-time ActiveX Controls are an important new feature introduced with Visual InterDev. They provide all of the benefits of component software offered by standard ActiveX Controls, such as plug-and-play functionality and visual editing at design time. However, design-time ActiveX Controls have no run-time component and generate HTML-based content, viewable on any platform and any browser.

Design-time ActiveX Controls can also automatically generate server-side or client-side scripting required to accomplish simple or complex tasks within a Web site. In essence, they are visual helper components that help a developer construct dynamic Web applications based on HTML.

It is hard to overstate the importance of design-time ActiveX Controls for Visual InterDev. The product derives maximum extensibility using design-time ActiveX Controls. Third-party software vendors and corporations can seamlessly extend the tool with specialized functionality using custom design-time ActiveX Controls. Design-time ActiveX Controls are based on the Component Object Model and hence can be shared across tools and applications.

Visual InterDev provides several integrated design-time ActiveX Controls, such as the Data Command Control for visually constructing complex data queries and other data access controls. These controls provide an easy-to-use visual mechanism for a developer who wants to construct complex database connectivity logic within a Web site. The control generates the HTML and scripting necessary for the run-time processing of the application that any standard Web browser can access.

Microsoft plans to deliver a growing set of design-time ActiveX Controls on the Visual InterDev Web site, in effect dynamically upgrading the tool with new functionality. In addition, Microsoft expects that over time, independent software vendors (ISVs) will offer hundreds of design-time ActiveX Controls compatible with Visual InterDev, the same way there are hundreds of ActiveX Controls available for use with Visual Basic.

A developer can build design-time ActiveX Controls using Visual Basic version 5.0 Control Creation Edition, Professional, and Enterprise, Visual C++, and other tools capable of creating standard ActiveX Controls. A design-time Software Development Kit (SDK) is available on the Microsoft Web site at www.microsoft.com/workshop/. Visual InterDev also supports standard ActiveX Controls as a way to extend Web pages with client-side software components that execute inside the browser.

Because design-time ActiveX controls differ from standard ActiveX controls in that they do not contain a binary, run-time component, Microsoft has named them as a separate category of ActiveX controls. It is important to note that design-time ActiveX controls do implement Component Object Model (COM) interfaces, so they can be arbitrarily shared across development tools provided by various software vendors. However, because they generate HTML and text-based scripting, there is no binary run-time component — hence their output can be viewed on any platform in any browser.

These controls are based on published COM interfaces developed during design previews held at Microsoft, and any interested third party can build them. Design-time ActiveX controls will co-exist with standard ActiveX controls, which do provide a binary run-time component that offers run-time functionality not offered by design-time ActiveX controls. For example, standard ActiveX controls, unlike design-time ActiveX controls, are exposed as COM objects at run time and design time, and can be scripted at run time by exposed methods, properties, and events.

Visual InterDev makes it easy to develop dynamic Internet and intranet applications based on Active Server Pages, a new feature for the Microsoft Internet Information Server.

As a server environment, Active Server Pages are compatible with the most popular Web browsers, so applications developed with Visual InterDev require no special client software and can be viewed through most browsers running any platform. To understand the flexibility of Visual InterDev to meet the needs of both Internet and intranet developers, it is important to first examine the rich functionality Active Server Pages provide.

Active Server Pages are a powerful new Web application development framework for building dynamic Web sites. Active Server Pages are HTML pages that contain scripts processed on the server before being sent to the Web browser running on the client computer. These server-side scripts, written in Visual Basic Script, JScript, Perl, or other scripting language, can make use of custom or packaged ActiveX Server Components to extend the Web server with application-specific functionality.

{ewc mvimg, mvimage,!illust.bmp}
Figure 4. Microsoft Internet Information Server and Active Server Pages.

Active Server Pages, formerly known as The ActiveX Server Framework, are planned as an integrated feature of Microsoft Internet Information Server version 3.0. Visual InterDev includes this feature, so existing users of the Microsoft Windows NT Server operating system 4.0 and Microsoft Internet Information Server 2.0 can easily upgrade their Web servers with this new functionality simply by installing the Visual InterDev server components on Windows NT Server 4.0. Active Server Pages can also be used with the development versions of Internet Information Server available on Windows NT Workstation 4.0 and Windows 95. In either case, there are no additional server licensing fees for this feature.

Originally, both the Web server and Web client computers could be called dumb, with a server sending HTML files, which rendered as formatted text to a client browser. This arrangement is still often in use. In this scenario, a user sends a request to a Web server by way of hypertext transport protocol (HTTP) for a particular HTML file. The server receives the request and sends the HTML file to the client's browser. The browser reads the HTML file and displays it accordingly. While this model provides instant access to nicely formatted pages of information for employees or potential customers, interaction between the user and the Web server computer is limited. In addition, the information is only as up-to-date as the most recent file posting on the Web server computer.

With Common Gateway Interface (CGI), Internet Server Application Program Interface (ISAPI), and other gateway interfaces, a user can send a request to an executable application using the HTTP protocol instead of requesting a static HTML file. In response, the server immediately runs the specified program. The program reads environment variables and standard input to determine the values passed with the request, such as values typed by a user to fill out an HTML form. The program then parses the values for meaningful information and generates HTML output to send back to the client computer for the user to read in a browser.

The disadvantage of gateway programs is that they can be difficult to create and change, and they require an entirely different design process than do HTML pages. In addition, CGI-type applications generate high server overhead since they require that the Web server change contexts between the server process and the CGI executable process for each program request.

Active Server Pages are based on the ActiveX Scripting engine for Microsoft Internet Information Server 3.0, which developers can use to include server-side executable scripts directly in HTML content. As a result, applications developed with Visual InterDev are:

- **Content-centric.** Completely integrated with the underlying HTML files.
- **Simple.** Easy to create with no necessity to compile or link programs.
- **Powerful.** Extensible with ActiveX Server Components.

What does this mean for Web application developers? It means the difference between publishing content and delivering interactive business applications, as these examples illustrate.

- For the human resources manager, it is the difference between publishing an HR handbook and offering interactive benefits management.
- For the real estate broker, it is the difference between advertising properties and delivering interactive appointment scheduling and mortgage application processing.
- For the travel agent, it is the difference between publishing airline schedules and delivering interactive online reservations and ticketing.
- For the mutual fund manager, it is the difference between publishing a prospectus and offering online portfolio management.
- For the IS professional, it is the difference between building a Web site to share documents and building a site to deliver fully interactive Web applications.

A developer can use various scripting languages, including Visual Basic Script and JScript, to build robust and highly scalable solutions using Active Server Pages. Visual InterDev offers a complete development system for developing applications with ActiveX Server Pages, including a variety of visual tools. Many of these tools automatically generate much of the server scripting necessary to perform complex tasks, such as establishing pooled database connections that can be used globally in a site; performing complex SQL queries against databases, and integrating the data into dynamically constructed HTML pages; and creating HTML forms that are directly linked to databases. The visual tools are powerful and easy to use, but for maximum flexibility Visual InterDev offers a developer the option of working directly with source code.

The basic ActiveX server unit is the Active Server Page file, or .ASP file. An .ASP file consists of ASCII text in one of three forms: text, HTML, or script.

This code is a complete, simple Active Server Page.

```
<HTML>
<P>
<%FORi=3 to 7%>
     <FONT SIZE=<%=i%>>
     Hello World!
     <BR>
     </FONT>
<%Next%>
</HTML>
```

Script consists of commands inside ActiveX script tags or HTML <SCRIPT> tags (or the inline equivalent <%> tags) that can be processed on the server by the ActiveX server engine. When a user requests a URL with an .ASP file extension, the ActiveX server engine reads through the file from top to bottom, executing any commands and sending text and HTML back to the user's browser. ActiveX Server scripting is completely integrated with HTML pages.

All that is necessary for the ActiveX Server engine to read a file is the presence of the .ASP file extension. Thus, existing HTML files can be renamed by removing the .HTML extension and adding an .ASP extension, making them Active Server Pages. Developers can then make the HTML text dynamic by adding ActiveX Server script to the .ASP page. The host language of ActiveX Server scripting is Visual Basic Script or JScript. However, a developer can use <SCRIPT> HTML tags to include scripts in the text built by other scripting languages, such as Perl.

In this example, server-side scripts are designated with an opening and closing <%> tag and are processed before the page is sent to the browser. In this case, a simple loop displays "Hello World" in an increasing font size within an HTML page. All server scripting is processed on the server, and only HTML is sent to the client. You can review the browser results in Figure 5.

{ewc mvimg, mvimage,!illust.bmp}
Figure 5.

The "Hello World" Active Server Page as displayed by Microsoft Internet Explorer together with the source HTML sent to the client computer as displayed by Notepad. The server-script is executed and then removed from the page by the server so that only the dynamic HTML is sent to the client.

Although writing ActiveX Server scripts with Visual InterDev is relatively easy, the scripts offer a powerful environment for Internet and intranet applications. For example, a developer can incorporate sophisticated functionality through ActiveX Server Components (previously known as OLE Automation Servers). ActiveX Server Components are an important feature of Visual InterDev. ActiveX Server Components provide an ideal way to extend the server-side functionality of Web applications with encapsulated, reusable business logic.

As COM components, Visual Basic Script and JScript can drive them and they can execute as either in-process DLLs or out-of-process executable files. For lightweight components designed as in-process DLLs, performance can be dramatically improved over CGI solutions, since no context switching between processes is incurred as Web users browse pages that use the component. In addition, ActiveX Server Components can be instantiated once and shared among multiple users connected to the Web site for more efficient use of server resources. Using Visual InterDev, developers can easily integrate ActiveX Server Components written in Visual J++, Visual Basic (version 4.0 or later), Visual FoxPro, Visual C, Visual C++, or other languages that can be used to build COM components.

Using Visual InterDev with ActiveX Server Components, a developer can easily create multi-tier Web applications. For example, a component that provides financial modeling and analysis functions can be used to build portions of a financial service Web application. Using Visual InterDev, a developer can create Active Server Pages that execute the component on the Web server and use the component's methods (functions) to return financial modeling information to users in dynamically constructed HTML pages. Because the component executes on the server, a user running any Web browser and using any platform can view the content.

ActiveX Server Components provide a convenient and effective way to tightly integrate a Web application with existing internal systems. For example, a client/server insurance-processing application written in Visual Basic can be exposed as a set of ActiveX Server Components that can be called directly from Web pages by way of ActiveX Server scripting. In this fashion, Visual InterDev helps to protect and extend the organization's investments in existing tools and systems, ranging from mainframe applications to more recently deployed client/server applications. Visual InterDev serves as the tool that can be used to effectively integrate the functionality of these systems directly into the Web site with ActiveX Server Components.

Using Visual InterDev, corporations can take advantage of distributed ActiveX Server Components with distributed COM (DCOM). DCOM offers complete network transparency, so developers program DCOM components using the same scripting logic as if the components were running locally on a Web server. DCOM components can provide greater scalability and fault tolerance for mission-critical Web applications.

{ewc mvimg, mvimage,!illust.bmp}
Figure 6.

Web application with distributed components. Active Server Components can be distributed via DCOM to provide greater scalability and fault tolerance for a Web application created using Visual InterDev.

Because ActiveX Server Components are COM objects, out-of-process server components can be seamlessly distributed over a server network using DCOM. This means that components requiring heavy processing can be distributed to application servers that work in conjunction with a Web server to efficiently process requests from users browsing the site. For example, a price look-up component that performs complex pricing calculations can be built as an ActiveX Server Component and distributed through DCOM to execute on a specialized application server.

One advantage of DCOM is that distributed computing is transparent to the developer. The ActiveX Server scripting used to execute and manipulate the remote component is the same as if the component were running directly on the Web server. Distributed solutions built with DCOM components and Visual InterDev offer more effective load balancing, higher performance applications, and greater fault tolerance for enterprise-class Web applications. Optionally, ActiveX Server Components can be managed using the Microsoft Distributed Transaction Server, which can be used to manage logical transactions among multiple components and also provides for additional scalability for distributed server-side Web components.

Click here to visit the Microsoft Distributed Transaction Server home page and read more about it:

{ewc mvimg, mvimage,!intjump.bmp}

Powerful database connectivity options and visual database tools are an integral component of Visual InterDev. The database connectivity features are based on the industry standard Open Database Connectivity (ODBC), and the visual tools work with any database supporting ODBC, including Oracle, Microsoft SQL Server, Microsoft Access, Microsoft Visual FoxPro, Informix, Sybase, IBM DB/2, and many others. In addition, using Visual InterDev, a developer can create scalable database solutions because it leverages Active Server Pages. The core database components of Visual InterDev include:

® Integrated Data View.

® Database Design-time ActiveX Controls.

® Database Wizards.

® The Query Designer.

® The Database Designer.

The integrated database access tools are based on Microsoft Active Data Objects, which provides flexible, object-based database connectivity to ODBC data sources. In addition, Visual InterDev can be used with alternate data access components developed by third parties.

{ewc mvimg, mvimage,!illust.bmp}
Figure 7. Inserting a data connection visually.

Visual InterDev offers a visual interface for quickly adding sophisticated database features to a Web site. Pictured above is the data connection dialog box a developer can use to add a data connection within a Web site to any ODBC data source. A developer uses the property page to visually set properties for the connection, such as the data source, cursor driver, and query time-out values. Visual InterDev performs the hard work of tagging the dynamic text with HTML and generating the server-side scripting, automatically saving the logic into an Active Server Page.

Active Data Objects (ADO) are used to provide flexible and scalable database connectivity within Visual InterDev applications. Optimized for Web-based data access, ADO offers an object-based approach to data access over the Web. Through ActiveX Scripting, connections to databases can be easily established to any ODBC data source, and a variety of methods within the component equip a developer with a powerful set of database commands for manipulating data and creating data-driven Web pages. Using ADO, a developer can manipulate database-defined data types, including binary large objects (BLOBs) such as GIF and JPG images retrieved from databases and those dynamically written into Web pages. In addition, ADO provides a rich set of properties for setting locking levels, cursor options, query and login time-outs, transaction support, result set scrolling, and error handling, to name a few. Using Visual Basic Script or JScript, ADO offers a developer maximum flexibility to develop powerful database functionality within a Web site.

Visual InterDev includes the Data View feature, which provides a visual interface to all of the databases being used within a Web site. Besides depicting each database connection being used in the site, the Data View offers a live connection to each database, so a developer can work directly with these databases within the IDE during Web site development.

For example, the developer can open any database to view tables, defined views, and stored procedures. The Data View can reveal detailed information about objects and properties within each database, including table definitions and field types, key structures, stored procedures, and so on. The Data View works with Query Designer and Database Designer features to deliver a sophisticated database development, administration, and maintenance system tightly integrated with the Visual InterDev IDE. The Data View works against any ODBC-compliant database and will show multiple connections against heterogeneous databases.

{ewc mvimg, mvimage,!illust.bmp}
Figure 8. The Visual InterDev Data View.

Visual InterDev includes an integrated view into any ODBC data source being used within the Web site. Pictured above, you can see the Data View pane on the left displaying an open data source to a database created with Microsoft Access and a Microsoft SQL Server database, including tables, fields, and views for each.

Visual InterDev builds on the ADO foundation by providing special design-time ActiveX Controls that automatically generate much of the server-side scripting, including the ADO calls necessary to establish database connections within a Web site, perform queries, and display results. In many cases, a developer can use the design-time ActiveX Controls to create data-driven Web sites with little additional programming required. For maximum flexibility, however, using Visual InterDev, a developer can view and develop directly in ActiveX Server scripting using the ADO component.

One example of a design-time control is the Microsoft Data Command Control. With the Data Command Control, a developer can select a database connection to an ODBC data source and then visually build a query against that connection using the Visual InterDev Query Designer. Once tested and complete, the Data Command Control generates all of the ActiveX Server scripting necessary to perform the query and automatically adds the scripting information to the appropriate Active Server Page. Data range controls offer similar functionality for visually building complex data queries, but also generate the looping and display logic necessary to page through a result set as a series of dynamically constructed HTML pages.

In addition to design-time ActiveX Controls, Visual InterDev also offers wizards that lead a developer through the process of creating custom data-bound HTML forms. After prompting a developer for information, the Data Form Wizard will automatically generate the HTML tags and ActiveX Server scripting information required to create complex HTML forms that are bound to databases.

For example, a developer can use the Data Form Wizard to visually construct a guest book form, complete with the logic to accept user input, update a database of registered users, and display a list of all users registered in the guest book database. Because the wizards, like design-time ActiveX Controls, generate standard HTML tags and ActiveX Server scripting, a developer can modify the generated source code as desired for further customization.

In addition, the Visual InterDev SDK is planned to be available in the first quarter of 1997. Using it, developers can build their own wizards using Visual Basic version 5.0, to further enhance Visual InterDev capabilities.

{ewc mvimg, mvimage,!illust.bmp}
Figure 9.

The Data Form Wizard steps a developer through the process of building sophisticated and highly customized HTML forms bound to databases. The wizard automatically generates the HTML tags and server-side scripting logic, which a developer can modify further if desired.

Visual InterDev offers a sophisticated Structured Query Language (SQL) Query Designer that works against any ODBC data source. The integrated Query Designer is available through an extremely easy-to-use interface for visually constructing even the most complex SQL statements, and generates the Data Manipulation Language (DML) for SELECT, INSERT, UPDATE, and DELETE queries. A developer can open live views of data sources and drag tables directly into a design pane of the Query Designer window to build a query. As a developer selects fields from tables, the SQL pane will display the dynamically constructed SQL statement. A developer can modify the SQL statement directly, and the changes will be reflected in the query design pane. In addition, a developer can execute any SQL statement to test it and display the results in a results pane. Finally, a developer can easily create complex queries across multiple tables, 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, execute arbitrary database definition language (DDL) commands directly against any ODBC data source, or perform ad hoc SQL queries. Visual InterDev thus delivers a complete, tightly integrated database development and administration tool for Web developers. It is important to note that the Query Designer works in conjunction with the Query Control (a design-time control). Once developed and tested in the Query Designer, the HTML tags and server scripting necessary to execute the query are generated automatically and embedded in the appropriate Active Server Page.

{ewc mvimg, mvimage,!illust.bmp}
Figure 10.

Using the Visual InterDev Query Designer, a developer can visually construct complex SQL queries against any ODBC data source.

In addition to offering the visual Query Designer, which works against any ODBC-compliant database, Visual InterDev also offers 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 create new databases with SQL Server and modify the structure and properties of existing databases in SQL.

Plus, database administration operations that used to take hours can be accomplished with a couple of clicks of the mouse. For example, a developer or database administrator can use the Database Designer to change the data type on a SQL field — for example, change a field from a type CHAR to a type INT — with a choice from a menu. Visual InterDev will then automatically change the field type. Ordinarily, this change would require 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. The Database Designer can generate DDL scripts that can be reviewed and submitted to database administrators for review and execution in controlled database environments.

The Database Designer can also be used to set up complex database designs with unique check constraints and to define relationships between tables using foreign keys. Because a developer can functionally group tables together, database diagrams can be dragged into the Query Designer to quickly construct queries against these logical groupings.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11.

Using the Database Designer, a developer can create database schemas for databases created with Microsoft SQL Server. The tool offers a powerful and flexible database-administration environment that simplifies the most complex administration tasks in SQL Server.

Some database tools for Web applications offer visual aids for easing the development of data-driven Web sites but do not deliver the scalability required for Internet sites or production intranet sites hosting mission-critical applications. Visual InterDev, however, delivers the scalability required for mission-critical, data-driven Web sites with both ease of use and extensive visual tools.

For example, global database connections can be established for an entire site and Internet Information Server will automatically pool these database connections across users. Pooling, connection caching, and time-out values are all established automatically based on default properties, but they can easily be customized by the developer. Visual InterDev also makes it easy to connect to multiple heterogeneous databases within a Web site, returning or updating data that has been visually integrated within a single HTML page for a user.

For example, Visual InterDev can be used to return data from an Oracle database running on a UNIX server as well as data from a database in SQL Server running on Windows NT, all integrated within a single HTML page for the user. In addition, based on ODBC support, a workgroup version of an intranet application can easily be designed on desktop databases, such as Microsoft Access or Visual FoxPro, and then can easily be scaled up to a server-based database, such as Microsoft SQL Server, without requiring complex code changes.

Visual InterDev includes extensive site management features integrated within the development environment. These features begin with the integrated project system that presents a live view of the site as it exists on the Web server. The project system offers a developer the ability to create new Web sites, view content on an existing Web site, and to create, add, or modify folders and files within a Web site. Site management features within the project system include:

- **Create a new Web site with the Web site wizard.** The wizard steps a user through the process of creating a new Web site, automatically creating a new site on a chosen Web server.
- **Manipulate a Web site file directly.** A developer can copy, delete, add, rename, and edit any file in the Web site from within the IDE, using the File View feature.
- **Import content files and directory trees into a Web site.** A developer can use Import File and Import Folder to import existing content from any local or network resource into the Web site.
- **Copy a Web site.** An entire Web site can be moved from one server to another server using Copy Web. This is useful for moving a Web site from a staging server to a production server, for example.
- **View any Web site object's properties.** A developer can right-click any object on the Web site, and Visual InterDev presents a property page full of useful information such as file size, modification date, and all incoming and outgoing links to a file.

The project system tracks links within a Web site to ensure they are valid. For example, when a developer renames a file, the system automatically detects all the pages throughout the site that reference that name as part of a URL reference (hyperlink), and automatically fixes these references to avoid broken links. In addition, should a file be moved or the site physically restructured using Visual InterDev, the system will automatically repair links in this manner. Finally, when a page or file is deleted, Visual InterDev can list all the references to that file that exist within the site.

Besides providing developers with automatic link repair and site management capabilities through the File View, Visual InterDev also has a powerful Web visualization tool called Link View. Link View presents a graphical view of the Web site, visually showing the logical relationship of pages within the site. For example, Link View can be used to visually depict the links from a particular HTML page to other pages inside or outside the Web site. A developer can expand or collapse the view to any level, showing as many pages or sections of the Web site as desired.

Besides diagramming links, Link View also depicts specific information on each file, using visual icons to depict its file type, such as HTML pages, Active Server Pages, .GIF or .JPG images, ActiveX Controls, Java applets, 2.5D Layout files, and design-time controls. Link View will graphically depict all broken links in red, as well as show external links in addition to internal links. Link View thus provides a smart design-time view of the site, and a developer can use the graphical view to launch editors on each file simply by double-clicking the file.

With Link View, a developer is also able to filter the view so that only specific information is displayed. A developer may choose to filter, in any combination, these Web site components:

® HTML pages.

® Multimedia files.

® Executable files.

® Documents.

® External files.

® Primary links.

® Secondary links.

{ewc mvimg, mvimage,!illust.bmp}
Figure 12.

As shown above, Link View displays a graphical representation of two HTML pages within a Web site and the references these pages have to various elements, such as .GIF images and .WAV files. Broken links are also shown.

Visual InterDev provides integrated source management for users of Microsoft Visual SourceSafe 5.0 version control software. Organizations using Visual SourceSafe can place a Web project under source control. Source management is especially helpful when teams work on the same Web site since it offers explicit check-in and check-out functions for all files in the site. In addition, the Visual SourceSafe revision tracking and merging features help developers to protect their work — offering, for example, the option to roll back to an earlier version of a file or to compare in detail the differences between two versions of a file.

Visual InterDev integrates with Visual SourceSafe so that any project can easily be put under source control without requiring that Visual SourceSafe be installed on a workstation of a developer who wants to use it. Instead, all Visual SourceSafe features execute from the server. When integrated with Visual SourceSafe, Visual InterDev projects can benefit from these Visual SourceSafe features:

- **Create Visual SourceSafe projects automatically.** When a project is placed under source control, Visual InterDev automatically creates the Visual SourceSafe project on a Web server and adds the existing content of the Web site to the source control database.
- **Check files in and out.** When a developer requests a Web site file for editing, Visual InterDev explicitly checks the file out of the Visual SourceSafe project before opening it on the developer's workstation. If the requested file is already checked out — for instance, another user of Visual InterDev or a user of FrontPage working on the same Web site — the developer is notified and offered a read-only copy.
- **Add elements to a Visual SourceSafe project automatically.** When a developer creates a new file for a Web site, Visual InterDev automatically puts this element under source control, using the Visual SourceSafe project set up for the site. For example, if a developer chooses to import a folder that contains 10 files and two subfolders, Visual InterDev automatically adds these elements to the Visual SourceSafe project. When a team member creates a new file, such as a new HTML page, Visual InterDev creates the new entry in the Visual SourceSafe project and automatically checks the element out to the developer.

Besides offering an integrated, visual environment for developing the next generation of server-based Web applications and data-driven Web sites, Visual InterDev also delivers integrated tools for developing Web pages, including Web pages with client-side scripting, ActiveX Components, and Java applets. In addition, Visual InterDev includes multimedia content editing tools: Microsoft Image Composer and Microsoft Music Producer. Using these tools, a developer can easily develop sounds and images for use in a Web site.

Visual InterDev includes a version of the award-winning FrontPage 97 HTML editor. Thus, teams of developers and business non-programmers can share a common, leading-edge HTML editing environment while using tools specifically designed to meet their differing needs.

Visual InterDev includes integrated support for adding ActiveX Controls, Java applets, and Netscape plug-ins to Web pages. When working with ActiveX Controls, a developer can use a visual Object Editor to visually modify a component. A visual property table, similar to the Visual Basic property table, offers a developer an easy way to modify a component's properties without ever needing to see or work with source code.

After working with a component visually, the Object Editor automatically generates all of the HTML syntax (based on the W3C industry-standard OBJECT tag) to bring the component into the Web page and execute it. Perhaps most importantly, the ActiveX Control is displayed in true WYSIWYG fashion directly in the FrontPage 97 editor and can even be dragged, dropped, and sized directly within the page. With more than 2,000 ActiveX Controls that deliver a wide range of functionality ranging from multimedia to multi-point conferencing to real-time data feeds, Visual InterDev offers true extensibility.

The integrated FrontPage editor also delivers support for easily inserting Java applets and Netscape plug-ins. A developer can visually set characteristics such as size and position, and the editor displays the frame for the Java applet or Netscape plug-in within the document.

Visual InterDev also includes a client-side Script Wizard, first introduced with the ActiveX Control Pad. The Script Wizard makes it easy to add interactivity to Web pages based on actions and events associated with ActiveX Controls. The Script Wizard includes a visual interface for easily connecting an event (such as a mouse click) with an action (such as playing a video clip). The Script Wizard lists all of the events associated with a given ActiveX Control in the Event pane. A developer can select an event in the Event pane and associate it with an action in the Action pane. Multiple actions can be added to a single event in this manner without any programming, since the Script Wizard generates the actual lines of script code added to the HTML page.

Using the Script Wizard, a developer can easily integrate the behaviors of multiple controls, thus creating an integrated application from potentially diverse sets of components, with each component providing a specific functionality. The Script Wizard generates either Visual Basic Script or JScript code, depending on developer preference. While aiding the developer in connecting events to client-side scripts, the Script Wizard code-entry window offers the developer a place to code in straight Visual Basic Script or JScript, if desired, or to modify the generated code. All client-side scripts developed with the Script Wizard are embedded as text within the HTML document using the W3C HTML `<SCRIPT>` tag.

{ewc mvimg, mvimage,!illust.bmp}
Figure 13.

Using the Script Wizard, a developer can employ a visual interface for building client-side scripts that wire together various ActiveX Controls on a Web page. Using the Event pane, a developer can select from a list of events and wire an event to an action in the Action pane.

Visual InterDev also includes a 2.5D, or frame-based, HTML Layout Editor introduced with the Microsoft ActiveX Control Pad. Using the HTML Layout Editor, which resembles a Visual Basic form, a developer can place multiple ActiveX Controls precisely, affording a developer exact control over x, y coordinate placement and z-ordering (layering) of controls. Rather than converting the form-based layout information into a best possible representation in HTML, (which currently offers no 2.5D layout capabilities) the HTML Layout Editor saves the information using new (draft) W3C syntax that extends HTML with 2.5D layout capabilities. This offers a developer exact control over object placement and object layering on a Web page. You can review the HTML Layout Editor in Figure 14.

{ewc mvimg, mvimage,!illust.bmp}
Figure 14. The HTML Layout Editor.

The HTML Layout regions use the new style sheet attributes specified in a W3C draft specification for Cascading Style Sheets and are rendered by the HTML Layout Control, which is part of the Microsoft Internet Explorer 3.0 Web browser. For more details about these attributes, see the Web site at www.w3.org/pub/WWW/TR/WD-style.

You can read more about the Microsoft implementation of 2.5D-style layout for HTML on the Web site at www.microsoft.com/workshop/author/layout/.

The HTML Layout Editor offers the following editing features:

- **Customize the Toolbox.** Right-click the Toolbox to add or delete individual controls supplied by third-party vendors.
- **Add custom tabs.** Add new Toolbox tabs using the Toolbox shortcut menu.
- **Build object templates.** Drag objects from the form to the Toolbox, creating an object template. The template captures all of the properties and can be used to easily incorporate objects with custom properties into new pages. Groups of controls can also be used as object templates in this manner.
- **Employ multiple levels of undo and redo.** Most edit operations can be easily undone or redone.
- **Automate control alignment, spacing, and sizing.** Align, size, and space controls automatically for easier layout.
- **Position controls using the drag-and-drop technique.** Position controls exactly with drag-and-drop editing. Controls can also be dragged from one page to another.
- **Set control layering (z-ordering).** Set object layering by right-clicking any control.
- **Employ the Script Wizard.** Choose the Script Wizard from the menu to add interactivity to a 2.5D layout region.

The HTML Layout Control offers run-time rendering of 2.5D layout regions within Microsoft Internet Explorer 3.0. It is included in the Complete Installation option of Microsoft Internet Explorer 3.0, and is also available for separate, automatic download as an ActiveX component. The HTML Layout Control also works in Netscape Navigator on Windows 95 and Windows NT, through the ActiveX plug-in for Netscape.

Microsoft Image Composer is especially useful for a developer who wants to modify or create images to fit a Web site and is designed as an easy-to-use tool for developers who are not graphics professionals. It recognizes most popular image formats, including directly reading Adobe PhotoShop files, while making it easy to save images into .GIF or .JPG formats for use on the Web. It is easy to arrange, customize, and create images in Microsoft Image Composer for use on Web sites developed and maintained in Microsoft Visual InterDev.

A developer launches Microsoft Image Composer from Visual InterDev by clicking a Toolbar icon or clicking images directly from within the IDE. The developer can use the Microsoft Image Composer command to save images directly to the Microsoft Visual InterDev project automatically.

Click here to visit the Microsoft Image Composer Web site for detailed information on this tool:
{ewc mvimg, mvimage,!intjump.bmp}

In addition to Microsoft Image Composer, Visual InterDev includes a unique tool for creating sound effects for use on Web sites. Music Producer employs the pioneering Microsoft Music Technology (MMT) and simplifies music creation with prebuilt, innovative musical styles. More versatile and powerful than static music clips, styles can define entire musical genres, such as samba or Texas swing, as well as musical moods or sets of instruments. More than 100 predefined styles are included. Even first-time users can create music immediately because everything needed to create quality original music is presented in one simple screen.

Microsoft Media Manager is a utility that makes it easy for a developer to track and manage dynamic and static media assets such as images, sound clips, video clips, HTML files, and office documents. It increases productivity because a developer can quickly search, preview, play, and retrieve media assets stored in Media Manager libraries. The utility integrates seamlessly with the Windows 95 and Windows NT operating systems and uses the same explorer-based navigation paradigm. Microsoft Media Manager offers extended search capabilities, thumbnail previews of images, and previews of other media content, such as sound clips and video clips, directly from the library explorer pane. A developer can add a media asset by simply using the drag-and-drop technique to drop it from a media library into a Visual InterDev project.

In this section, you will learn about the different data access models that are available, the object hierarchy provided by ADO, and how ActiveX Data Objects (ADO) communicate with a database.

This section includes the following topics:

By Shamir Dasgupta
Published by The Cobb Group

If you're a Web developer and you want to take advantage of Microsoft Internet Explorer's special features, you might opt for creating a page that uses Internet Explorer-specific tags. However, a page that uses these tags won't always look good when it's viewed with other browsers, such as Netscape Navigator or Mosaic. For example, if you use the tag <MARQUEE> to scroll the text on your page, the text will appear static when viewed with Netscape Navigator.

Likewise, if you work with multimedia and animation, you can easily enhance your work with an ActiveX Control, and IE users can view your content without having to download a plug-in. Unfortunately, Navigator users can't view your ActiveX controls or certain other Internet Explorer-specific features.

It might seem that you're stuck with an unwelcome choice: Either avoid using IE-specific features altogether, or accept the fact that your page won't look its best when viewed with another browser. But you don't have to settle for either of these options.

To please a variety of users, you can create distinctive pages specific to mainstream browsers such as Internet Explorer and Netscape Navigator. In this article, we'll show you how to determine whether the user is using Microsoft Internet Explorer and, if so, how to display an Internet Explorer-specific page.

Advantage: Explorer

Microsoft Internet Explorer (IE) is the most versatile browser available today. At the time of this writing, it's also the only browser that supports W3C's HTML 3.2 standard. In addition to supporting Java, IE supports Microsoft's ActiveX Object Model. And since it's an OLE-compliant product, it can integrate with other products in the Microsoft Office family.

Creating Separate Versions

If you create two distinct versions of your page, you'll be able to automatically direct viewers to one or the other based on their browser type. To optimize your Web page for each user's browser, you'll build a main page that contains a browser-detection script. (We'll use JavaScript, since both Internet Explorer and Netscape Navigator support it.) The script will route IE users directly to an IE-enhanced version of the page and Netscape Navigator users to a different version. The two graphics in **Figure A** show examples of a page enhanced for IE and Navigator, respectively.

{ewc mvimg, mvimage,!illust.bmp}
{ewc mvimg, mvimage,!illust.bmp}
Figure A. By creating two versions of your page, you're able to direct viewers based on their browser type.

To make the transition seamless, the main page itself won't ever appear in either browser. However, users whose browser doesn't support JavaScript *will* view the main page, which will contain its own version of your page's content.

JavaScript is a simple but powerful object-based scripting language. Unlike Java programming language, JavaScript isn't compiled — it's *interpreted* by the browser. To implement JavaScript in an HTML document, you'll use a <SCRIPT> tag, such as this:

```
<SCRIPT LANGUAGE="JavaScript">
<!--hide script
...your script goes here
//-->
</SCRIPT>
```

You'll notice that we use the HTML comment tag (<!--) to hide the script from browsers that don't support scripting.

Object of Desire

In our example, we'll take advantage of JavaScript's object-oriented nature. *Objects* are the basic building blocks of an object-oriented language, and JavaScript is no exception. The top level of objects in JavaScript

consists of those objects belonging to the navigator (not necessarily Netscape, but the browser). The *navigator* object has four properties that supply the values: *appName*, *appVersion*, *appCodeName*, and *userAgent*. We're especially interested in *appName*, because it returns the name of the user's browser. For example, if the user is working in Internet Explorer, the property *appName* will return the value *Microsoft Internet Explorer*. We can then redirect the browser to an Internet Explorer-enhanced version of our page.

Making It Work

As we mentioned, we want to implement the script in the main page, which we'll name INDEX.htm. Besides the script, we'll also need to add text for the browsers that don't support scripting. To do so, use an HTML editor to add the script in **Listing A**.

Listing A. Source code for INDEX.htm

```
<SCRIPT LANGUAGE="JavaScript">
<!--
var name = navigator.appName;

function isBrowser()
{

  if (name == "Microsoft Internet Explorer")
  {
  location.href="INDEX1.htm"
  }

else

  {
  location.href="INDEX2.htm"
  }
}

isBrowser()
// -->
</SCRIPT>
<BODY>
Content for the users not using either browser.
</BODY>
```

Note that we're redirecting the Internet Explorer users to INDEX1.htm and the Navigator users to INDEX2.htm. You should place this script between the <HEAD> and <BODY> tags so that the script executes before you load the content.

Shamir Dasgupta is editor-in-chief of *Microsoft Web Builder*, The Cobb Group's journal for Web developers. His personal Website, www.xprssions.com, won three categories, including Grand Prize, in Microsoft's inaugural *Activate the Internet* competition in 1996.

By Shamir Dasgupta
Published by The Cobb Group

Online commerce — it's one of the fastest-growing uses for the Net. Both Microsoft and Netscape have contributed to this growth by creating specialized Web server software for secure online transactions. If you're developing for a small company, however, you may be unable to dedicate a server for online commerce. You could create your own online transaction system using traditional Common Gateway Interface (CGI) scripting in Perl or a heavy-duty programming language such as C, but you'd still need support from your Internet Service Provider. This support can be expensive — if it's available at all.

JavaScript may be the answer to this growing dilemma. In this two-part series, we'll show you how to build an inexpensive online transaction system using JavaScript. In this article, we'll build the order selection part of our application, which the client side will handle. In Part II, we'll tackle building the server-side order form and the transport section.

Object-based Scripting

JavaScript is an object-based scripting language designed to extend the capabilities of the Web. Unlike Sun Microsystems' Java, which is a fully compiled language, JavaScript is interpreted by the browser.

By design, JavaScript can manipulate and present information through the browser, but it can't retrieve information from another file or save data to the Web server or the user's computer. Likewise, you can't use JavaScript to write a program that will scan the user's directory and retrieve or erase the user's files. Since JavaScript can't alter the user's system, it's a safe choice for client-side scripting. Currently both Microsoft Internet Explorer 3.x and Netscape Navigator 2.01 or higher support the JavaScript language.

If You Build It, They Will Come

For our purposes, we'll build a fictitious online store called "L.L. Cool Bean." We'll pretend this is an online catalog store, where viewers can order casual clothing such as shirts, sweaters, and similar apparel.

In this example, your users can select one of three styles of L.L. Cool Bean shirts — Bay, Rugby, or Active — and then choose options such as color, size, finish, and sleeve style. Some of these options cost extra. After making a selection, the user can opt to calculate the total cost of the shirt. If everything is satisfactory, the user then clicks a button and sends all the order information directly to the store for processing.

As we mentioned, you don't need a dedicated server for this application. We'll show you how to handle the product selection within the client side. We'll use dropdown boxes for the Shirt, Finish, and Sleeve Style options and radio buttons for the Color and Size selections. We'll use FORM elements to create dropdown boxes and radio buttons and JavaScript to generate them dynamically. When we've finished, our online store will look like the one shown in **Figure A**.

Now get started by firing up your favorite HTML editor. We used Microsoft ActiveX Control Pad to create our pages.

{ewc mvimg, mvimage,!illust.bmp}
Figure A. You can use JavaScript to create an online store without having to use a dedicated server.

Working with Forms

When you're designing a Web page that will use forms, it's much easier to design the forms in standard HTML and then add the scripting. Forms aren't very straightforward, and trying to correct problems in the form design while simultaneously debugging the script makes the whole process more difficult than necessary. For this demonstration, we could simply create straight code for dropdown boxes and radio buttons using the <FORM> tag, as shown here:

```
<form name="LLCoolBean">
    <select name="MySelection">
    <option value=0>-select-a-style-
    <option value=1>Bay Shirt
    <option value=2>Rugby Shirt
    <option value=3>Active Shirt
```

```
    </select>

</form>
```

However, the store's inventory may change frequently. For example, the store may want to sell different styles of shirts or add more color options. For every change, you'd have to add the items in more than one place in the script, which could become a maintenance nightmare. To solve that problem, you can use JavaScript to generate the list of products dynamically, based on a "database" of components. Instead of using an external database, though, you'll build custom arrays to hold your information.

What's an Array?

An *array* is a structure that can hold multiple pieces of the same type of data (strings, integers, floats, or objects, for example). JavaScript arrays can be *associative* in nature, which means you can associate a particular location within the script with certain information or data. To create a custom array for our page, you'll need to first declare the object. The following shows the basic code for a custom array:

```
function MakeArray(size) {
   this.length = size;

   for(i = 1; i <= size; i++)
      this[i] = null;

   return this;
}
```

Using this technique for your online store, you can build an "array of options" that will hold the name and cost of each option as a separate element. For example, the code shown below for the function named *Options()* creates an *Options* object, with a *name* and a *cost*.

```
function MakeArray(size) {
   this.length = size;

   for(i = 1; i <= size; i++)
      this[i] = null;

   return this;
}

function Options(strName, iCost) {
   this.name   = strName;
   this.cost   = iCost;
}
```

For our form, you'll need to create each of the five components of the shirts with its own array of options and costs, as described in **Listing A**, which you'll find at the end of this article.

I'm sure by now you're beginning to see the advantage of using the array database. To add a new option, all you have to do is change the number specified by *MakeArray()* and add an additional *baseList[?] = new Options(…)* line.

These array databases will permit JavaScript to dynamically build the form that will display the products to your potential buyers. Next, we'll need to create the script that will dynamically generate the input fields.

Creating Options

You can take advantage of JavaScript's object-oriented nature by creating a generic function that will build the option list for your users with all the arrays we've built so far. A single generic function can handle all the arrays because, to JavaScript, an object is an object, no matter what type of object it is. In the code in **Listing A**, each of the different *Options* arrays has the same structure: a *length* property, and *name* and *cost* properties for each

index element. In the following code, we'll generate the *createSelect()* function to build a dropdown list box based on the information in that array:

```
function createSelect(strName, objList) {
   var tStr = "<SELECT NAME=\"" + strName + "\">"
            + "<OPTION SELECTED>- select-a-style -";

   for(var i=1; i<=objList.length; i++)
      tStr += "<OPTION>" + objList[i].name;

   tStr += "</SELECT>";

   return tStr;
}
```

Later in the body of the HTML document, simply add JavaScript's *document.write* function to generate the <SELECT> tag. Using a similar technique, write a *createRadio()* function for creating the Radio Button <INPUT> tag shown here:

```
function createRadio(strName, objList) {
   var tStr = "";

   for(var i=1; i<=objList.length; i++) {
      tStr += "<INPUT NAME=\"" + strName + "\" "
            + "TYPE=radio VALUE=\"" + objList[i].name
            + "\">" + objList[i].name;
   }

   return tStr;
}
```

Keeping a Running Tally

Since different costs are associated with different options, we'll need to keep a running tally of each option for our calculation. As you can see in **Listing B** at the end of this article, you'll need to create a separate calculation function for each option.

Notice that each function has a basic structure:

1. The selected item
2. A warning message if no item is selected
3. The cost added to the running tally

The warning message will ensure that the store receives only valid orders. If the user fails to select an option, the application will display an alert box, as shown in **Figure B**. Our next step will be to calculate the total cost from the running tally.

{ewc mvimg, mvimage,!illust.bmp}
Figure B. We've used alert boxes to warn users that they have neglected to select an option.

Calculating Total Cost

To calculate the total cost, simply call each separate cost function and total them when the user clicks on the Calculate The Cost button. The code in **Listing C** at the end of this document shows the *calculate()* function.

Creating the HTML Body

So far, we've created the script to handle the form components and calculate the cost of each option. Now we're ready to build the order selection form itself. We'll use JavaScript to create the body of the document and to generate the <SELECT> and <INPUT> tags for the form. To do so, we'll use JavaScript's *document.write*

function as shown in **Listing D**. (Notice that the listing contains only the code that appears between the <BODY> and </BODY> tags.)

We've included some additional code for formatting purposes — the finished page will look like the one shown in **Figure A**, above. When the user makes all necessary selections and clicks the Calculate The Cost button, the application will display a summary of the user's selections, along with the total cost, as shown in **Figure C**.

{ewc mvimg, mvimage,!illust.bmp}
Figure C. The application will provide the user with a summary of the selected item and the total cost.

Conclusion

Online commerce is the future of the Web. In Part I of this series, we've provided some ideas for you to use as you begin building your own online store. We've shown how to create the selection form and calculate a running cost using JavaScript. In Part II, we'll create a server-side order form and the transport program, which will enable users to send their selections to the online store.

Listing A. Creating the list of options

```
baseList        = new MakeArray(3);
baseList[1]     = new Options("Bay Shirt - $42", 42);
baseList[2]     = new Options("Rugby Shirt - $52" , 52);
baseList[3]     = new Options("Active Shirt - $60", 60);

colorList     = new MakeArray(3);
colorList[1]  = new Options("Sky Blue", 0);
colorList[2]  = new Options("Windows Blue - add $5", 5);
colorList[3]  = new Options("Midnight Blue", 0);

sizeList      = new MakeArray(4);
sizeList[1]   = new Options("Small", 0);
sizeList[2]   = new Options("Medium", 0);
sizeList[3]   = new Options("Large", 0);
sizeList[4]   = new Options("X- Large - add $5", 5);

washList      = new MakeArray(2);
washList[1]     = new Options("Stone Washed", 0);
washList[2]     = new Options("Soft Washed - add $5",5);

sleeveList    = new MakeArray(2);
sleeveList[1]   = new Options("Short Sleeve", 0);
sleeveList[2]   = new Options("Long Sleeve", 0);
```

Listing B. Cost function for each option

```
function calcBase() {
   var which = document.LLCoolBean.base.selectedIndex;

   if(which == 0) {
      alert("\nYou must select a shirt.\n");
      return false;
   }

   totalCost += eval(baseList[which].cost);
   strDesc   += " - " + baseList[which].name + "\n"
       + " L.L.Cool Bean Quality\n" + " Unlimited Exchange\n"
       + " Friendly Service\n";

   return true;
}
```

```
function calccolors() {
   var which = 0, j = -1;

   for(var i=0; i<document.LLCoolBean.colors.length; i++) {
      if(document.LLCoolBean.colors[i].checked) {
         j = i;
         break;
      }
   }

   if(j == -1) {
      alert("\nYou must select a color " + "for your shirt.\n");
      return false;
   }

   for(var i=1; i<=colorList.length; i++) {
      if(document.LLCoolBean.colors[j].value == colorList[i].name) {
         which = i;
      }
   }

   totalCost += eval(colorList[which].cost);
   strDesc   += " - " + colorList[which].name + " \n";

   return true;
}

function calcsize() {
   var which = 0, j = -1;

   for(var i=0; i<document.LLCoolBean.size.length; i++) {
      if(document.LLCoolBean.size[i].checked) {
         j = i;
         break;
      }
   }

   if(j == -1) {
      alert("\nYou must select a size " + "for your shirt.\n");
      return false;
   }

   for(var i=1; i<=sizeList.length; i++) {
      if(document.LLCoolBean.size[j].value == sizeList[i].name) {
         which = i;

      }
   }

   totalCost += eval(sizeList[which].cost);
   strDesc   += " - " + sizeList[which].name + "\n";

   return true;
}

function calcwash() {
```

```
   var which = document.LLCoolBean.wash.selectedIndex;

   if(which == 0) {
      alert("\nYou must select a finish" + "for your shirt.\n");
      return false;
   }

   totalCost += eval(washList[which].cost);
   strDesc   += " - " + washList[which].name + "\n";

   return true;
}

function calcsleeve() {
   var which = document.LLCoolBean.sleeve.selectedIndex;

   if(which == 0) {
      alert("\nYou must select a sleeve" + "style for your shirt.\n");
      return false;
   }

   totalCost += eval(sleeveList[which].cost);
   strDesc   += " - " + sleeveList[which].name + "\n";

   return true;
}
```

Listing C. The calculate() function

```
function calculate() {
   totalCost   = 0;
   strDesc     = "";

   if(!calcBase())
      return false;

   if(!calccolors())
      return false;

   if(!calcsize())
      return false;

   if(!calcwash())
      return false;

   if(!calcsleeve())
      return false;

   alert("\nYou have selected the following " +
         "L.L.Cool Bean Shirt:\n\n" + strDesc +
         "\n with a total cost of $" + totalCost + ".");

   return true;
}
```

Listing D. HTML code for the document's body

```
<BODY bgcolor="#0080c0">
```

```
<CENTER>
<font face="arial" size=2 color=#ffffff>
<H3>Welcome to L.L.Cool Bean</H3>
<H1> The Online Store</H1>
<BR>
<BR>
<FONT COLOR=#0080c0>
<BR>
<FORM NAME="LLCoolBean" METHOD=POST>
<TABLE WIDTH=600 CELLPADDING=4 BORDER=0 BGCOLOR=#a8e2ff>
 <TR>
    <TH VALIGN=TOP>L.L.Cool Bean's Shirt:<BR>
    <FONT SIZE=1>
     Select from three <BR>
     distinctive styles<BR>
     of genuine Cool Bean <BR>
     shirts.</FONT>
    </TH>
    <TD>
      <SCRIPT LANGUAGE="JavaScript">
      <!-- begin hide
        document.write(createSelect("base", baseList) + "<BR>");
      // end hide -->
      </SCRIPT>
      L.L.Cool Bean's Shirts.<BR>
      Exceptionally Soft,<BR>
      Stone or Soft Washed,<BR>
      All-Cotton Knit.<BR>
      The softest colors and textures<BR>
      we've ever offered.
    </TD>
  </TR>
  <TR>
    <TH VALIGN=TOP>Available Colors:</TH>
    <TD>
      <SCRIPT LANGUAGE="JavaScript">
      <!-- begin hide
        document.write(createRadio("colors", colorList) + "<BR>");
      // end hide -->
      </SCRIPT>
    </TD>
  </TR>
  <TR>
    <TH VALIGN=TOP>Available Sizes:</TH>
    <TD>
      <SCRIPT LANGUAGE="JavaScript">
      <!-- begin hide
        document.write(createRadio("size", sizeList) + "<BR>");
      // end hide -->
      </SCRIPT>
    </TD>
  </TR>
  <TR>
    <TH VALIGN=TOP>Soft or Stone Washed:</TH>
    <TD>
      <SCRIPT LANGUAGE="JavaScript">
      <!-- begin hide
```

```
        document.write(createSelect("wash", washList) + "<BR>");
      // end hide -->
      </SCRIPT>
    </TD>
  </TR>
  <TR>
    <TH VALIGN=TOP>Long or Short Sleeves:</TH>
    <TD>
      <SCRIPT LANGUAGE="JavaScript">
      <!-- begin hide
        document.write(createSelect("sleeve", sleeveList) + "<BR>");
      // end hide -->
      </SCRIPT>
    </TD>
  </TR>
</FONT>
<TR>
<TD>
<BR>
<BR>
<INPUT TYPE=button NAME="compute"
       VALUE="Calculate the Cost" ONCLICK="calculate()">
</TD>
</TR>
</TABLE>
</FORM>
</FONT>
</CENTER>
</BODY>
```

Shamir Dasgupta is editor-in-chief of *Microsfot Web Builder*, The Cobb Group's journal for Web developers. His personal Website, www.xprssions.com, won three categories, including Grand Prize, in Microsoft's inaugural *Activate the Internet* competition in 1996.

By Shamir Dasgupta
Published by The Cobb Group

If you have a product to sell, chances are you've considered trying to sell it over the Web. But many would-be Web retailers have been stymied by questions of transaction security, the inability to dedicate a server to online commerce, or the lack of affordable support from ISPs.

In this series, we show you how to bypass all these difficulties by building your own online transaction system. In Part I, we created the first part of an online clothing store using JavaScript. We built an order selection form where your potential customers can select from a list of shirt types, color, finish style, and sleeve style. We also created a calculation method that keeps a running total of all the options the user selects; with the click of a button, we can display a message box containing the total amount of the order and the options the user has selected. **Figure A** shows the order selection form, and **Figure B** shows the message box.

{ewc mvimg, mvimage,!illust.bmp}
Figure A. Our online store looked like this at the end of Part I.

{ewc mvimg, mvimage,!illust.bmp}
Figure B. This message box summarizes the customer's selections.

Now we need to find a way to transmit this information, along with the customer's name and address, to the store for processing. Once again, we'll turn to JavaScript to accomplish our goals.

Customer Information

With one minor modification, we'll change our existing message box into a confirmation box, as shown in **Figure C**. If the customer clicks OK, our application will generate a customer information form, as shown in **Figure D**, where the customer can fill in his or her name and address. Clicking the Submit Order button sends the order information to the store.

{ewc mvimg, mvimage,!illust.bmp}
Figure C. Now the customer can verify the cost and place the order by clicking OK.

{ewc mvimg, mvimage,!illust.bmp}
Figure D. The customer will fill out name and address information in this form.

To make all this happen, we'll first build a function that will generate our customer information form and specify the mode of transport we'll be using. Next, we'll modify our existing application to ensure that the customer completes the order selection form correctly. Finally, we'll modify the existing message box into the confirmation box shown in **Figure C**.

Adding a New Function

Let's get started by creating a new function that will dynamically generate the customer information form. To do so, add the code shown in **Listing A**, at the end of this document, to our original code from Part I. As you can see, we've created a new function called *order()* and used JavaScript's *document.write* function to create the form elements.

We'll need to add the item's cost and the description to the form so it can be submitted to the store for processing. However, we don't want to display the fields twice. Notice that we've added two hidden fields where we're storing the cost and description.

Transport Options

Depending on your Internet Service Provider (ISP), you'll need to modify the line

```
document.write("<FORM name='order2' METHOD=POST>");
```

If your ISP provides you with its mail transport, you can simply add an ACTION parameter and the name of the transport program, like this:

```
document.write("<FORM name='order2' METHOD=POST ACTION=MyServer/mail.exe>");
```

If you don't have access to the mail transport program, you can have customers submit the form via mail, like this:

document.write("<FORM name='order2' METHOD=POST
ACTION='mailto:user@somewhere.com'>");

Using the MAILTO tag simply brings up the user's default E-mail program and embeds the information from the form in the body of the mail document. While this method works, it doesn't look very sophisticated. We suggest you use the mail transport provided by your ISP if possible.

Completing the Order

Next, we need to modify our existing application so it will generate the customer information form only when the customer makes a complete selection. To do so, we'll modify the HTML code for the order selection form's Calculate The Cost button, so it reads:

```
<INPUT TYPE=button NAME="compute"
    VALUE="Calculate the Cost" ONCLICK="if(calculate()) order();">
```

For our final step, we'll modify the *alert()* feature of the *calculate()* function to display a Confirm dialog box, so the customer can verify the order one last time. Replace the current code, shown below:

```
alert("\nYou have selected the following " +
         "L.L.Cool Bean Shirt:\n\n" + strDesc +
         "\n with a total cost of $" +
         totalCost + ".");
```

with the following code to complete the modification:

```
return confirm("\nYou have selected the following " +
         "L.L.Cool Bean Shirt:\n\n" + strDesc +
         "\n with a total cost of $" +
         totalCost + ".\n" +
         "\n Do you want to order now?");
```

At this point we've finished adding and modifying the code and are now ready to try out our new online store. Save the changes to the file and open it in Internet Explorer. This time, when you click on the order selection form's Calculate The Cost button, it will display the confirmation dialog box shown in **Figure C**. If you click OK, you'll be presented with our customer information form, as shown in **Figure D**. Clicking on the Submit Order button will send the order information to the store. Congratulations! You're now open and ready for business.

Listing A. Code for order()

```
function order() {
   document.write("<body bgcolor=#ffffff>");
   document.write("<h4>Thank you for shopping at L.L.Cool Bean!</h4>");
   document.write("<FORM name='order2' METHOD=POST>");
   document.write("First Name: <INPUT NAME='first'  TYPE=TEXT> ");
   document.write("Last Name: <INPUT NAME='last' TYPE=TEXT><br>");
   document.write("Street Address: <INPUT NAME='street' TYPE=TEXT
            åsize=40><br>");
   document.write("City: <INPUT NAME='city' TYPE=TEXT> ");
   document.write("State: <INPUT NAME='state' TYPE=TEXT size=2> ");
   document.write("Zip Code: <INPUT NAME='zip' TYPE=TEXT size=5>");
   document.write("Phone: <INPUT NAME='phone' TYPE=TEXT><br><br>");
   document.write("You have selected the following L.L.Cool Bean Shirt - "
                  + "which will cost $" +
                  + totalCost + ":"
                  + "<PRE>" + strDesc + "</PRE>");
   document.write("<INPUT value='Submit your Order' TYPE=SUBMIT><br>");
   document.write("<INPUT NAME='description' TYPE=HIDDEN>");
   document.write("<INPUT NAME='cost' TYPE=HIDDEN></form></body>");

   document.order2.description.value = strDesc;
```

```
   document.order2.cost.value = totalCost;
}
```

Shamir Dasgupta is editor-in-chief of *Microsfot Web Builder*, The Cobb Group's journal for Web developers. His personal Website, www.xprssions.com, won three categories, including Grand Prize, in Microsoft's inaugural *Activate the Internet* competition in 1996.

By Shamir Dasgupta
Published by The Cobb Group

In recent years, the Internet has become the medium of choice for many creative media publishers. Industry pundits estimate that a whopping 65 percent of all media publishers maintain an online presence. As more sophisticated tools for developing Web content become available, the number of interactive Web sites will continue to grow.

If you happen to be a painter or photographer or simply a collector, one way to display your masterpieces to the world is to make them available on the Internet. In this article, we'll show you how to build an interactive online gallery using ActiveX Controls and Visual Basic Script.

The Active Gallery

For our example, we'll create a page with two simple images and a toggle button. The first image, as shown in **Figure A**, is a photograph taken inside an art gallery. Our second image, shown in **Figure B**, is the same picture, but in this case we've applied a spotlight effect using Adobe PhotoShop.

{ewc mvimg, mvimage,!illust.bmp}
Figure A. In our first image, we didn't use any special effects.

{ewc mvimg, mvimage,!illust.bmp}
Figure B. We used a spotlight effect on this image.

Rather than displaying both images on the page at once, we want to add some user interaction so users can switch between the two pictures using a toggle button. We also want the button's caption to change when the user clicks on it. So, we'll display the first image and the toggle button when the page loads. The initial caption will advise users to click the button to turn on the spotlight, as shown in **Figure C**. When the user clicks the button, the second image will appear, and the caption will change to tell users to click the button to turn the spotlight off.

{ewc mvimg, mvimage,!illust.bmp}
Figure C. Our ActiveX Control displays the first image with the toggle button to turn on the spotlight.

Although we've selected a dramatic effect for our example, you may find a more practical use for this trick. For example, if you're selling your house and want to put pictures of it online, you can let your viewers toggle between views of the front and back of the house. Or you can offer both a normal and a close-up view of the front yard — the possibilities are limitless.

Building a Gallery From the Ground Up

When you're working with images and buttons and you want them lined up properly on a page, you should create an HTML Layout (ALX) file. Using this method gives you precise control over placement of your objects. In our example, the objects we'll place in our Layout file will be ActiveX Controls: an Image control and a ToggleButton control. Once we've placed the controls, we'll create the interaction between them using Visual Basic Script. Finally, we'll create an HTML page to house the Layout file.

Getting Started

To get our project underway, we'll create our HTML Layout file. The best way to do so is to use the ActiveX Control Pad from Microsoft.

Start Control Pad by selecting it from Windows' Start menu. Select New from the File menu, highlight HTML Layout from the New dialog box, and click OK. Control Pad will display a blank Layout page and the Toolbox dialog box. This Toolbox holds all the ActiveX Controls installed in your system. When you move your mouse pointer over the control icons, a tool tip will display the name of each control in turn.

Click and drag the Image control to the layout grid, and then double-click on the control to bring up its properties. To match our image sizes, enter the values *300* and *200* for the Width and Height, respectively. Enter the name of the first image, including the extension, under the PicturePath property. This image will be our default picture.

Click on the X icon to close the Properties dialog box. The layout will display the first image. Next, click and drag the ToggleButton control to the layout grid, and position it under the Image Control. Now change the caption to

Welcome to the Active Gallery. Click here to turn on the spotlight over the painting. When you're finished, your HTML Layout should look like the one in **Figure D**. Save the file as GALLERY.alx.

{ewc mvimg, mvimage,!illust.bmp}
Figure D. HTML Layout gives you precise control over your object placement.

Now that we have our basic controls in place, we're ready to write the script. If you want to work from scratch, you can use VBScript or JavaScript to create the interaction between the ActiveX Controls. If you're new to scripting, the Control Pad's built-in Script Wizard can help you get started. However, the quickest way to duplicate our example is to just copy our VBScript code from the beginning of **Listing A**. (We've highlighted in color the code you need to enter.)

To insert the script, right-click on the Layout and select View Source Code. You'll be able to see actual source code for the ActiveX Controls on the layout. Now, place the script anywhere in the page and save the changes to the file.

Sprucing Up

If you like, you can do some cosmetic work on the Layout file at this point. (Depending on your version of the Control Pad, you may have to retrieve the Layout file by selecting Open from the File menu.) Right-click on the Layout file to bring up the Properties dialog box. In this example, we've changed the background color to Black and added a TextBox control to display the words *Active Gallery*. Feel free to experiment with different design effects; after all, it's your creative display! **Listing A** at the bottom of this document shows the complete code for our example.

The Final Step

We've just finished creating the HTML Layout file containing the ActiveX Controls and the script. Now we'll need to create an HTML page to house our HTML Layout. To do so, start a new HTML document in the Control Pad. Select Insert HTML Layout from the Edit menu, then highlight our file, *GALLERY.alx*, and click OK. That's all there is to it! The Control Pad does the rest. As you'll see, the Control Pad will insert the necessary code to include the ALX file in the document. Your code should look like this:

```
<HTML>
<HEAD>
<TITLE>Active Gallery Demo</TITLE>
</HEAD>
<BODY bgcolor=#000000 text=#ffffff>

<OBJECT CLASSID="CLSID:812AE312-8B8E-11CF-93C8-00AA00C08FDF"
ID="gallery_alx" STYLE="LEFT:0;TOP:0">
<PARAM NAME="ALXPATH" REF VALUE="file:C:\My Documents\mit\gallery.alx">
</OBJECT>

</BODY>
</HTML>
```

Notice that we've added code to make the background color of the page match the background color of our Layout Control. Now, save the file as ACTIVE_GALLERY.htm and open it in Internet Explorer. When you do, it should look like **Figure C**. Click on the toggle button, and you'll notice that the picture as well as the caption changes, as you can see in **Figure E**.

{ewc mvimg, mvimage,!illust.bmp}
Figure E. When you click on the toggle button, you'll see the second image.

Listing A. Complete code for gallery.alx

```
<SCRIPT LANGUAGE="VBScript">
<!--
Sub ToggleButton1_Click()
If ToggleButton1 then
   Image1.picturepath = "light.bmp"
```

```
   ToggleButton1.caption = "Welcome to the Active Gallery." & _
      "Click here to turn on the spotlight over the painting."
else
   Image1.picturepath = "spotlight.bmp"
   ToggleButton1.caption = "Thank you for visiting the " & _
      "Active Gallery. Click here to turn off " & _
      "the spotlight when you leave."
end if

end sub
-->
</SCRIPT>
<DIV BACKGROUND="#0" ID="Layout3" STYLE="LAYOUT:FIXED;WIDTH:400pt;
      HEIGHT:300pt;">
    <OBJECT ID="Image1"
            CLASSID="CLSID:D4A97620-8E8F-11CF-93CD-00AA00C08FDF"
            STYLE="TOP:74pt;LEFT:86pt;WIDTH:225pt;HEIGHT:150pt;ZINDEX:0;">
        <PARAM NAME="PicturePath" VALUE="c:\my documents\mit\light.bmp">
        <PARAM NAME="BorderStyle" VALUE="0">
        <PARAM NAME="SizeMode" VALUE="3">
        <PARAM NAME="Size" VALUE="7938;5292">
        <PARAM NAME="PictureAlignment" VALUE="0">
        <PARAM NAME="VariousPropertyBits" VALUE="19">
    </OBJECT>
    <OBJECT ID="ToggleButton1"
     CLASSID="CLSID:8BD21D60-EC42-11CE-9E0D-00AA006002F3"
STYLE="TOP:248pt;LEFT:102pt;WIDTH:192pt;HEIGHT:40pt;TABINDEX:0;ZINDEX:1;">
        <PARAM NAME="BackColor" VALUE="2147483663">
        <PARAM NAME="ForeColor" VALUE="16711680">
        <PARAM NAME="DisplayStyle" VALUE="6">
        <PARAM NAME="Size" VALUE="6773;1411">
        <PARAM NAME="Value" VALUE="True">
        <PARAM NAME="Caption" VALUE="Welcome to the Active Gallery.
                 Click here to turn on the spotlight over the painting.">
        <PARAM NAME="FontEffects" VALUE="1073741825">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="ParagraphAlign" VALUE="3">
        <PARAM NAME="FontWeight" VALUE="700">
    </OBJECT>
    <OBJECT ID="TextBox1"
     CLASSID="CLSID:8BD21D10-EC42-11CE-9E0D-00AA006002F3"

STYLE="TOP:25pt;LEFT:91pt;WIDTH:215pt;HEIGHT:25pt;TABINDEX:1;ZINDEX:2;">
        <PARAM NAME="VariousPropertyBits" VALUE="746604571">
        <PARAM NAME="BackColor" VALUE="16711680">
        <PARAM NAME="ForeColor" VALUE="16777215">
        <PARAM NAME="Size" VALUE="7568;873">
        <PARAM NAME="Value" VALUE="Active Gallery">
        <PARAM NAME="BorderColor" VALUE="8421504">
        <PARAM NAME="SpecialEffect" VALUE="1">
        <PARAM NAME="FontEffects" VALUE="1073741825">
        <PARAM NAME="FontHeight" VALUE="360">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="ParagraphAlign" VALUE="3">
        <PARAM NAME="FontWeight" VALUE="700">
    </OBJECT>
```

</DIV>

Shamir Dasgupta is editor-in-chief of *Microsfot Web Builder*, The Cobb Group's journal for Web developers. His personal Website, www.xprssions.com, won three categories, including Grand Prize, in Microsoft's inaugural *Activate the Internet* competition in 1996.

By Shamir Dasgupta
Published by The Cobb Group

Some user actions can be very frustrating to Web developers. For example, you may include a registration form on your site that you want users to complete. However, they may leave out some key data or enter invalid values in certain fields. You can prevent this from happening by validating the form data. Then, if users leave a required field blank or enter an invalid value, your program can alert them.

Traditionally, you implement form field validation on a Common Gateway Interface (CGI) script and process it on the server. While this process is effective, it's not very efficient. In this article, we'll show you how to validate your form data on the client (browser) side using JavaScript.

Introducing JavaScript

JavaScript is an object-based scripting language you can use to develop client and Internet applications. JavaScript isn't related to the Java programming language that Sun Microsystems developed; however, you can call a Java applet from JavaScript. Microsoft Internet Explorer 3.0's ActiveX Scripting Model supports JavaScript. JavaScript is interpreted code and has the following advantages: It's object-based, event-driven, secure, and platform-independent. Let's review exactly what these terms mean.

Unlike object-oriented languages, objects are built into object-based languages such as JavaScript. For example, you don't need to build a Time object in JavaScript because it's built in.

JavaScript is designed to react when a browser event occurs. For instance, you can write a script that instructs the browser to navigate to a different location if an onClick event occurs.

JavaScript can't retrieve or save data in the client or the server — it can only *manipulate* the data. Therefore, a user's computer and server are safe from harmful content, such as viruses.

You can view a JavaScript-enabled page on any machine, regardless of the operating system on which it's running. As long as your target audience is browsing with a JavaScript-enabled browser, such as Internet Explorer 3.0 or higher or Netscape Navigator 2.0 or higher, they will be able to view and interact with your program.

Our Example

In our example, we'll show you how to verify that the user has completed each required field value on a form. We use a typical registration form, which asks for information regarding the user's name, street address, E-mail address, and so on. We want our JavaScript to ensure that the user completes each field before submitting the form. (Our example doesn't include the Perl script that transports the form to the server after the users click the Submit button.)

JavaScript Implementation

You can include JavaScript in your HTML document by using the <SCRIPT>…</SCRIPT> tag, as shown:

```
<script language="JavaScript">
...your code goes here
</script>
```

Currently, both Microsoft Internet Explorer 3.0 and higher and Netscape Navigator 2.0 and higher support JavaScript. To hide your code from those browsers that aren't compatible with JavaScript, use the HTML comment tag. For example, the code

```
<script language="JavaScript">
<!-- hide your code
...your code goes here
//  end hide-->
</script>
```

will hide the JavaScript code from incompatible browsers. Please note that the ending comment tag begins with a double forward slash (//). Because this line falls within JavaScript domain, you must tell the interpreter that it's

a comment. (All comment lines start with a double forward slash in JavaScript.)

Form Elements

The registration form we've used for our example is fairly simple. In the form shown in **Figure A**, we simply ask the user to enter her name, street address, telephone number, and E-mail address. The code we used for our form appears in **Listing A**, which you'll find at the bottom of this document.

{ewc mvimg, mvimage,!illust.bmp}
Figure A. The Registration Form asks users to complete registration information.

In this code, we call a function named *checkData* when the user submits the form. In the next section, we'll create this function using JavaScript.

What to Validate

In our form, we've defined the first name, last name, street address, telephone number, and E-mail address fields as text objects. A text object is normally an input field on an HTML form. However, by using JavaScript, you can utilize the text object to display a value that the browser returns to the viewer.

For example, we want the browser to display an alert message if the user leaves the FirstName text object empty. To create this warning, you simply write

```
if (document.forms[0].FirstName.value.length ==0)
{alert("Please enter your first name.")
 return false}
```

After you do so, the script will display an alert box that reads *Please enter your first name* if the user attempts to submit the form before completing the First Name field.

Ensuring that the name fields aren't empty is pretty straightforward. On the other hand, verifying the user's E-mail address requires a little more coding. You must check the entry's known E-mail address characters (such as @) to be sure they're valid. For example, the following code will ensure that the user is entering the E-mail address in a valid format:

```
if (document.forms[0].Email.value.length >0){
   i=document.forms[0].Email.value.indexOf("@")
   j=document.forms[0].Email.value.indexOf(".",i)
   k=document.forms[0].Email.value.indexOf(",")
   kk=document.forms[0].Email.value.indexOf(" ")
   jj=document.forms[0].Email.value.lastIndexOf(".")+1
   len=document.forms[0].Email.value.length
```

Putting It All Together

In the form, we refer to the checkData function, so you want to check that the script loads before the form does. For this example, we placed the script before our HTML page's <HEAD> tag. Your finished script should look like the code in **Listing B** (located at the bottom of this document).

Save the document as *REGISTRATION.HTM* and open it in Internet Explorer. To test the script, properly complete all the fields — except the E-mail field, in which you should enter the invalid value *bill*. When you do so, the JavaScript will display the alert box shown in **Figure B**. For the complete code listing of this HTML page, refer to **Listing C** at the bottom of this document.

{ewc mvimg, mvimage,!illust.bmp}
Figure B. You can use JavaScript to validate your form fields.

Listing A.

```
<HTML>
<HEAD>
<TITLE>Registration</TITLE>
```

```
</HEAD>
<BODY>

<h1>Registration Form</h1>
<br>
<FORM METHOD=POST ACTION=FormProgram.pl  onSubmit="return checkData()">

<pre>

First Name  <input type=text size=32    maxlength=50  name="FirstName">
<br>
Last Name   <input type=text size=32    maxlength=50  name="LastName">
<br>
Address <input type=text size=32    maxlength=255 name="Address1">
<br>
City    <input type=text size=32    maxlength=50  name="City">
<br>
State   <input type=text size=32 maxlength=50  name="State">
<br>
Zip Code    <input type=text size=12    maxlength=12  name="ZipCode">
<br>
Telephone   <input type=text size=23    maxlength=30 name="Phone">
<br>
E-mail  <input type=text size=23    maxlength=50 name="Email">
<br>
<br>
<INPUT TYPE="SUBMIT" VALUE="Register">
<INPUT TYPE="RESET" VALUE="reset">

</pre>
</FORM>
</BODY>
</HTML>
```

Listing B.

```
<SCRIPT language="JavaScript">
<!--
function checkData (){

 if (document.forms[0].FirstName.value.length ==0){
  alert("Please enter your first name.")
  return false}
 if (document.forms[0].LastName.value.length ==0){
  alert("Please enter your last name.")
  return false}
 if (document.forms[0].Address1.value.length ==0){
  alert("Please enter an address.")
  return false}
 if (document.forms[0].City.value.length ==0){
  alert("Please enter a city.")
  return false}
 if (document.forms[0].State.value.length ==0){
  alert("Please enter a State.")
  return false}
 if (document.forms[0].ZipCode.value.length ==0){
  alert("Please enter a valid ZipCode.")
  return false}
```

```
 if (document.forms[0].Email.value.length >0){
   i=document.forms[0].Email.value.indexOf("@")
   j=document.forms[0].Email.value.indexOf(".",i)
   k=document.forms[0].Email.value.indexOf(",")
   kk=document.forms[0].Email.value.indexOf(" ")
   jj=document.forms[0].Email.value.lastIndexOf(".")+1
   len=document.forms[0].Email.value.length

   if ((i>0) && (j>(i+1)) && (k==-1) && (kk==-1) && (len-jj >=2) && (len-jj<=3)) {
     }
   else {
     alert("Please enter an exact email address.\n" +
     document.forms[0].Email.value + " is invalid.")
     return false}}
}
//-->
</SCRIPT>
```

Listing C.

```
<HTML>
<SCRIPT language="JavaScript">
<!--
function checkData (){

 if (document.forms[0].FirstName.value.length ==0){
  alert("Please enter your first name.")
  return false}
 if (document.forms[0].LastName.value.length ==0){
  alert("Please enter your last name.")
  return false}
 if (document.forms[0].Address1.value.length ==0){
  alert("Please enter an address.")
  return false}
 if (document.forms[0].City.value.length ==0){
  alert("Please enter a city.")
  return false}
 if (document.forms[0].State.value.length ==0){
  alert("Please enter a State.")
  return false}
 if (document.forms[0].ZipCode.value.length ==0){
  alert("Please enter a valid ZipCode.")
  return false}
 if (document.forms[0].Email.value.length >0){
   i=document.forms[0].Email.value.indexOf("@")
   j=document.forms[0].Email.value.indexOf(".",i)
   k=document.forms[0].Email.value.indexOf(",")
   kk=document.forms[0].Email.value.indexOf(" ")
   jj=document.forms[0].Email.value.lastIndexOf(".")+1
   len=document.forms[0].Email.value.length

   if ((i>0) && (j>(i+1)) && (k==-1) && (kk==-1) && (len-jj >=2) && (len-jj<=3)) {
     }
   else {
     alert("Please enter an exact email address.\n" +
     document.forms[0].Email.value + " is invalid.")
     return false}}
}
```

```
//-->
</SCRIPT>
<HEAD>
<TITLE>Registration</TITLE>
</HEAD>
<BODY>

<h1>Registration Form</h1>
<br>
<FORM METHOD=POST ACTION=FormProgram.pl onSubmit="return checkData()">

<pre>

First Name  <input type=text size=32 maxlength=50 name="FirstName">
<br>
Last Name   <input type=text size=32 maxlength=50 name="LastName">
<br>
Address <input type=text size=32 maxlength=255 name="Address1">
<br>
City    <input type=text size=32 maxlength=50 name="City">
<br>
State   <input type=text size=32 maxlength=50 name="State">
<br>
Zip Code    <input type=text size=12 maxlength=5 name="ZipCode">
<br>
Telephone   <input type=text size=23 maxlength=30 name="Phone">
<br>
E-mail      <input type=text size=23 maxlength=50 name="Email">
<br>
<br>
<INPUT TYPE="SUBMIT" VALUE="Register">  <INPUT TYPE="RESET" VALUE="reset">

</pre>
</FORM>
</BODY>
</HTML>
```

Shamir Dasgupta is editor-in-chief of *Microsfot Web Builder*, The Cobb Group's journal for Web developers. His personal Website, www.xprssions.com, won three categories, including Grand Prize, in Microsoft's inaugural *Activate the Internet* competition in 1996.

What are the main differences between using IDC/HTX, Active Server Pages, and Advanced Data Connector for accessing databases over the Web?

IDC (Internet Database Connector)/HTX is a feature of Microsoft Internet Information Server (IIS) that you can use to write HTML templates that can be filled with the results of ODBC queries.

With IIS 3.0, the combination of Active Server Pages and ActiveX Data Objects (ADO) supercedes this technology. By using Active Server Pages with ADO, you can construct scripted HTML pages and use ADO inside the script to access databases and fill the HTML page with the results.

Both of these approaches are useful for scenarios where client intelligence is either not desirable or not available. Both yield a completely static client experience, and the data is not updatable. For example, the client might receive a text table, but a user can't manipulate that table.

Advanced Data Connector (ADC) is a complementary approach. It uses object invocations to get data onto the client computer, where it can be programmatically manipulated and updated. Data can also be bound to ActiveX controls. ADC achieves a richer user experience, but currently only works in Internet Explorer 3.0 or later.

What is the caching limit on the client? What are the other limitations of the current release?

® Client-side caching is limited only by the available virtual memory on the client computer.

® Currently, only recordsets returned as a return value of a method call can be marshaled to the client over HTTP. There is no support to marshal recordsets returned as out parameters or as more than one recordset. Note that this limitation is not a limitation when you use the DCOM protocol.

® Business objects on the middle tier do not retain their state between invocations.

® The **ADOR.Recordset** object on the client is not updatable and has a limited programming model.

® You need to release rowsets in the client-side cache after you make update method calls.

® Advanced Data Connector 1.0 uses an all-or-nothing update programming model if you use the **AdvancedDataFactory** object's **SubmitChanges** method to apply updates.

What is the support for DB transactions? Can I use Business Objects that are running under MTS with ADC 1.0?

® ADC 1.0 uses Optimistic Concurrency control (based on timestamps, where available) for submitting updates.

® You can implement server-side business objects to run under Microsoft Transaction Server and hence participate in distributed transactions. You can invoke these objects from the client side through the **AdvancedDataSpace** object just like non-Transaction Server objects. However, you can't use the **AdvancedDataSpace** object to coordinate Transaction Server transactions from the client side. (All coordination takes place on the server; the **AdvancedDataSpace** object only invokes the objects that may be part of a transaction.)

Why does Advanced Data Connector use OLE DB over ODBC instead of direct ODBC API to access data?

This allows Advanced Data Connector to work with datasets without providers (that is, remoted recordsets). Also, this way Advanced Data Connector can load a cache with data from non-SQL data sources.

What data types are supported for marshaling?

Advanced Data Connector 1.0 supports OLE Automation data types (except safe arrays and variants) and **ADOR.Recordset** and **ADODB.Recordset** data types.

Does Advanced Data Connector support CORBA for invoking business objects on the middle tier?

Advanced Data Connector 1.0 doesn't currently support this.

Can Advanced Data Connector run in Netscape Navigator 3.0 or on non-Internet Information

Server Web servers?

Advanced Data Connector 1.0 doesn't currently support this.

Can I use ActiveX Control Pad, Visual InterDev, or FrontPage to build Advanced Data Connector applications?

Advanced Data Connector 1.0 doesn't currently support these.

Are there any wizards to help end users create Advanced Data Connector applications or to help port existing Visual Basic applications to the Web?

Advanced Data Connector 1.0 doesn't currently provide any wizards for this.

Can I use parameters in ADO Command objects to specify my queries at run time? Is there any such analogy on the client side?

No. Advanced Data Connector doesn't support the complete ADO programming model on the client; it supports only the **Recordset** object on the client. The only way to specify the query is by concatenating user inputs into a SQL string. On the middle tier, however, you can use the complete ADO programming model.

How can I hide the user ID and password in my connection string so that a user cannot view them (via View Source) in Internet Explorer?

You can hide the user ID and password by establishing access from Internet Information Server to SQL Server via a trusted connection. You can configure this DSN option on the IIS computer. If the option is set, the connection is established by using the IUSR_<machinename> account (the anonymous IIS user) and the Windows NT authentication used to connect to SQL Server. You can then restrict your CONNECT property to use only the DSN parameter, and omit the UID and PWD parameters altogether.

Can I use Advanced Data Connector components in a Visual Basic application? I am particularly interested in invoking Automation objects over HTTP from within my Visual Basic application. I am also interested in using the advanced ADC recordset marshaling features over the DCOM protocol.

Yes. You can use Advanced Data Connector components within a Visual Basic application. You can use the **AdvancedDataSpace** object within your Visual Basic object to create a client-side proxy for the Automation objects that can be accessed over HTTP. The following code shows how to do this:

```
Private Sub Command2_Click()
    Dim ads As Object
    Set ads = CreateObject("AdvancedDataSpace")

    Dim bo As Object

    Set bo = ads.CreateObject("ProgIdOfAutomationObject", _
        "http://<ServerName>")
    Set rs = bo.MethodNameThatReturnsARecordset()
    While Not rs.EOF
        Print rs(0)
        rs.MoveNext
    Wend
End Sub
```

For invoking the same object over DCOM, you have to first mark the object as "safe for scripting" and "safe for initialization" on your client computer. When you use DCOM, your server-side object must be compiled as an .exe file; if you want to use a .dll file, then it must be running under Microsoft Transaction Server. (For more information on setting up DLLs to run under Microsoft Transaction Server, see the readme.txt file. The readme.txt file also contains details on how to mark an object as "safe for scripting and initialization.") In the client application, you need to make one change in the **AdvancedDataSpace** object **CreateObject** method call:

```
Set bo = ads.CreateObject("<ProgIdOfAutomationObject", _
    "<ServerName>")
```

This allows you to use the recordset marshaling over DCOM and also gives you the advantage of working with a client-side cache. This makes scrolling through rows in the recordset faster and reduces the number of network roundtrips.

Advanced Data Connector samples don't work. More specifically, I get the "Unexpected Error (80040e21)" message when I try to run Advanced Data Connector

General Suggestions

Server Checklist

® Make sure that you have the latest (released) version of Internet Information Server (IIS) version 3.0 with Active Server Pages installed on your server computer. This installs the latest ADO and OLE DB components that Advanced Data Connector needs.

The version and timestamps of the ADO and OLE DB DLLs. They are installed under the C:\Program Files\Common Files\System directory. You may have the wrong (Beta) version of IIS. The correct versions and timestamps are:

ADO		
Msader10.dll	1.00.994	11/25/96
msado10.dll	1.00.994	11/25/96
msador10.dll	1.00.995	12/6/96

OLEDB:		
msdadc.dll	01.10.2326	11/26/96
msdaenum.dll	01.10.2326	11/26/96
msdaer.dll	01.10.2326	11/26/96
msdaerr.dll	01.10.2326	11/26/96
msdasql.dll	01.10.2326	11/26/96
msdasqlr.dll	01.10.2326	11/26/96
msdatl.dll	01.10.2326	11/26/96
msdatt.dll	01.10.2326	11/26/96

All of these should be installed with the release version of IIS 3.0, which is available at http://www.microsoft.com/iis/default.asp.

The correct version and timestamps for the Advanced Data Connector DLLs are:

MSADC:		
MSADC10.dll	1.00.1015	10/15/96
msadcb10.dll	1.00.1211.0	12/11/96
msadcc10.dll	1.00.1211.0	12/11/96
msadcF10.dll	1.00.1211.0	12/11/96
msadco10.dll	1.00.1211.0	12/11/96
msadcS10.dll	1.00.1211.0	12/11/96

® If you re-install IIS, you must re-install ADC.

® If you are using Microsoft SQL Server as your back-end data source, use Microsoft SQL Server 6.5, Service Pack 2.

® Ensure that the ODBC DSNs are System DSNs and not User DSNs. The Anonymous user logging on through IIS can view only System DSNs.

® Make sure that the password authentication for your WWW service allows anonymous access. Also check that the anonymous account it uses is a user on your domain that has NTFS permissions to all the relevant servers and directories. Lastly, check that the password for this anonymous account has not expired.

® Since Advanced Data Connector is installed in the C:\Program Files\Common Files\System\MSADC directory, check that the Anonymous user from the IIS Service Manager can access this directory. It should have both **Read** and **Execute** access on this directory.

® If you have trouble with Microsoft Advanced Data Connector DLLs and sample applications, re-register them by using Regsvr32.exe.

® If you are writing your own business objects, make sure that you set the registry entries so that the objects can be launched in IIS. (For information on setting up these registry entries, see the Readme.txt file.) Place your Active Server Pages script files (.asp) in a directory that gives the Anonymous user (in IIS) execute permission.

Client Checklist

® Make sure that the security level in Microsoft Internet Explorer is set to **Medium** or **None** (**None** not recommended).

Customer Ranking Demo Application

The URL referencing Richtx32.ocx is incorrect. You must copy Richtx32.ocx from the C:\Program Files\Common Files\System\MSADC\Samples\RankDemo directory to the C:\Program Files\Common Files\System\MSADC\ samples directory for the application to be able to use this control.

Address Book Sample Application

Make sure a "Guest" login exists for the SQL Server where the sample table has been installed. Make sure that the Guest user ID has sufficient rights to read and write the sample table. The Address Book application assumes the password for the Guest user is "guest".

Miscellaneous

® If your application needs to support updatability, make sure that you have configured the ActiveX controls on your Web pages to be updatable. Use the following syntax:

```
<PARAM NAME="ParamName" VALUE="Value">
```

For example, the following code appears between the OBJECT tags that define a grid control, and allows users to update data displayed in the grid control:

```
<Param Name="AllowUpdate" Value="True">
```

® Advanced Data Connector can work across HTTP and DCOM. VBScript and JScript clients can use HTTP and COM to call methods on Automation objects registered on servers. When objects are invoked only over the HTTP protocol, type libraries for the server objects are not needed on the client.

® There are three kinds of bound controls in Microsoft Visual Basic: simple controls, simple data-aware controls, and complex data-aware controls.

Advanced Data Connector does not distinguish between simple controls and simple data-aware controls. It recognizes controls as either accepting a value or accepting a whole cursor. For a simple control, the binding syntax is:

<control>.<property>=<column>;

For example:

```
richtext1.text=coltext;
```

This causes Advanced Data Connector to put the current value for <column> into the <property> of the <control>. For a complex control, you only need to specify the name of the control to tell Advanced Data Connector to hand it the cursor. For example:

```
Grid1;
```

For more details on the binding grammar you can use in Advanced Data Connector, search the Microsoft Advanced Data Connector 1.0 (msadc10.hlp) for “context-free grammar.”

In this section, you will learn how to use IIS and Windows NT to control access to files in your Web site by setting permissions.

You can use IIS to set Read and Execute permissions on virtual directories in your Web site. You can also set permissions on files and folders by using Windows NT File System (NTFS).

This section includes the following topics.

You should place your Web pages and data files on an NTFS partition. When you use an NTFS partition, you can set permission for users or groups of users on individual files and folders.

u **To set permissions on files and folders**

1. From Windows NT Explorer, right-click the file or directory for which you want to set permissions and click **Properties**.
2. On the **Security** tab, click **Permissions**.
3. In the **Permissions** dialog box, add or remove users and specify the type of access allowed for each user.

To see an illustration of the **Security** tab, click this icon.
{ewc mvimg, mvimage,!illust.bmp}

You can use a combination of IIS settings, NTFS permissions, and server-side script to protect your Web pages.

For example, if you want to allow all users to access most of your Web site — but restrict a few pages to certain users — you can configure your Web site as follows:

® In Internet Service Manager, double-click the WWW service, then from the **Service** tab, click **Allow Anonymous**, **Basic Authentication**, and **Windows NT Challenge/Response**.

® For the pages you want to restrict, use Windows NTFS file permissions to remove the Anonymous account IUSR_*computername* from the access list, and then add the users for whom you want to allow access.

When a user requests a Web page, the user is logged on as anonymous and IIS attempts to access the Web page. If access is denied, the user will then be prompted for a login ID and password. Only users who provide a valid login ID and password will be able to access the restricted page.

You can control what actions a Web user can perform on your database. For example, you may want to allow some users to read and update tables and limit other users to read-only access.

When a Web user requests an Active Server Page that connects to a Microsoft SQL Server database on a separate computer running Windows NT, the user is first logged onto the other computer running Windows NT and then onto the SQL Server.

This illustration shows the logon process from a Web browser to an SQL Server.

{ewc MVIMG, MVIMAGE,!W10G030.BMP}

The following steps list the logon process that occurs when a user requests an Active Server Page that connects to an SQL Server database.

1. If the Web server allows anonymous logon, the user is mapped to the anonymous account, for example, IUSR_test.
2. The .asp file runs script to connect to an SQL Server database as shown in this example.

   ```
   Set conn = Server.CreateObject("ADODB.Connection")
   conn.Open "DSN=StateU;UID=SA;PWD=;"
   ```

3. For the .asp file to connect to the SQL Server database, the user is first logged on to the Windows NT-based computer where SQL Server is installed.
4. Finally, the user is logged onto the SQL Server itself.

Logging onto the Windows NT-Based Computer

IIS will attempt to log onto the Windows NT-based computer with the user's current account. For example, if the user is mapped to the IUSR_test account, the system will log onto the other Windows NT-based computer as IUSR_test.

If this logon attempt fails, the system will attempt to log on as Guest.

For the Web user to access the SQL Server database, the Windows NT-based computer on which SQL Server resides must either create the appropriate Windows NT account (for example IUSR_test) or enable the Windows NT Guest account.
{ewc mvimg, mvimage,!tip.bmp}

Logging on to SQL Server

The login ID for SQL Server is provided in the connection string argument when opening a connection with ActiveX data objects. The Web user does not need to know or supply an SQL Server login ID. The login ID and password can be provided in the server-side script.

The SQL Server login ID must have the appropriate rights defined within SQL Server to access any requested tables.

If you disable anonymous logon on your Web server so that all users must provide a login ID, you can write ASP script that retrieves the login information provided by a browser. You can then create different connections to a database depending on the login ID provided.

This code sample sets the SQL Server login ID to "SA" or "ReadOnlyUser" depending on the user account provided by the browser.

```
login = Request.ServerVariables("LOGON_USER")
If strcomp (login, "server\jane",1) = 0 Then
    cstring = "DSN=pubs;UID=sa;pwd"
Else
    cstring = "DSN=pubs;UID=ReadOnlyUser;pwd"
End If
Set conn = Server.CreateObject("ADODB.Connection")
conn.Open cstring
```

If you prevent anonymous logon, each request made to your Web server must include a valid Windows NT logon ID and password. To obtain this information, the browser will prompt the user for an account name and password.

To see a demonstration that requests a Web page that requires a valid Windows NT logon, click this icon. {ewc mvimg, mvimage,!democlip.bmp}

To prevent anonymous logon

1. In Internet Service Manager, double-click the WWW service to display its property sheet, then click the **Service** tab.
2. Clear the **Allow Anonymous** check box, then click **Basic**, **Windows NT Challenge/Response**, or both, and click OK.

Authentication Methods

You can specify how the logon information is sent from the Web browser to the Web server by setting the authentication method.

Basic

If you select Basic authentication, the user name and password is sent from the Web browser to the Web server as plain text. This method is useful because it is generally supported by all Web browsers; however, it is less secure than the Windows NT Challenge/Response method.

Windows NT Challenge/Response

If you select Windows NT Challenge/Response, the user name and password is encrypted during transmission from the Web browser to the Web server. This method of authentication is supported by Microsoft Internet Explorer version 2.*x* and later.

Using Both Basic and Windows NT Challenge/Response

If you select both Basic and Windows NT Challenge/Response authentication, Windows NT Challenge/Response will be used if it is supported by the browser. Otherwise, Basic will be used.

If you select all three methods of authentication (anonymous, Basic, and Windows NT Challenge/Response), every request to a Web page will attempt to access the page as anonymous. If the request fails, the user will be prompted for a login ID and the request will be attempted again.

Using Logon Information in ASP Script

If the browser sends a logon ID to the Web server, you can write server-side script to retrieve this information from the **ServerVariables** collection of the **Request** object using this syntax:

Request.**ServerVariables** (*variable*)

Values for the variable argument can include the following:

® LOGON_USER

Returns the login ID provided by the browser.

® AUTH_TYPE

Returns the authentication type used by the browser.

You can use this information to customize the Web page for different users as shown in this code sample:

```
<%
If strcomp (Request.ServerVariables("LOGON_USER"), _
            "myusername", 1) = 0 Then
    Response.Redirect "privatepage.htm"
Else
    Response.Redirect "publicpage.htm"
```

```
End If
%>
```

Active Server Pages provides a server-side scripting environment that you can use to create interactive Web applications.

For information about creating Active Server Pages, see Authoring Active Server Pages in Chapter 2: Developing a Web Project.

Server-Side Scripting

Server-side scripts run on the Web server rather than on the client. The Web server does all the work involved in processing and generating the HTML pages that are returned to the Web browser.

Any controls or components you use in your server-side script must exist and run on the Web server.

Creating an Active Server Page

An Active Server Page is a file with the extension .asp. An .asp file is a text file that can contain any combination of the following elements:

® Text

® HTML tags

® Server-side script

When a user views an .asp file, the Web server processes any server-side script contained in an .asp file, and returns HTML to the browser.

The following example code shows how to use an Active Server Page that displays a different greeting to a user, depending on the time.

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

<HTML>
<BODY>

<% If Hour(Now) < 12 Then %>
Good Morning! <BR>
<% Else %>
Good Day! <BR>
<% End If%>
Welcome to the Home Page.
</BODY>
</HTML>
```

You could also display the greeting by writing a separate function and explicitly calling that function from the Active Server Page. For example:

```
<%@ LANGUAGE="VBSCRIPT" %>
<%
Function Greeting()
    If Hour(Now) < 12 Then
        Greeting = "Good Morning!"
    Else
        Greeting = "Good Day!"
    End If
End Function
%>
<HTML>
<BODY>

<%= Greeting() %>
Welcome to the Home Page.
```

```
</BODY>
</HTML>
```

Note There is no debugging environment for Active Server Pages.

By David Chappell
Briefly described in Books Available from Microsoft Press
Understanding ActiveX and OLE
Published by Microsoft Press

Chapter 11: ActiveX, the Internet, and the World Wide Web

From its modest beginnings as a U.S. government-sponsored research network, the Internet has developed into a genuine phenomenon. By providing a global network linking millions of computers, the Internet makes possible things that once weren't even conceivable. And as ever-increasing numbers of homes and offices set up high-speed connections to this network, we can expect still more advances that are today inconceivable. The availability of cheap bandwidth — and the ubiquitous global network it makes possible — might prove to be a technical innovation as transforming as the invention of the microprocessor.

Like most new hardware-oriented innovations, the Internet expanded so rapidly because of a "killer" application, attractive enough to motivate people to use it. That killer app was the World Wide Web. The Web today is a major source of information and commerce for millions of people around the world. Web technology has found a receptive home in the business world, too, as corporate intranets based on Internet technologies have proliferated rapidly. Using these technologies, private organizations can build their own internal webs, allowing them to share information inside an organization just as the Internet-based Web does externally. With its easy-to-use, easy-to-understand user interface, web technology has broad appeal.

COM is fundamentally about defining the boundaries between pieces of software. The Internet has a major impact on those boundaries in several ways. The Web's browsing metaphor also affects how applications interact both with data and with their users, two more traditional concerns for technologies built using COM. To address these changes, several new COM-based technologies have been created, and others have been adapted for this new environment. This chapter explores these new and adapted technologies.

This chapter includes the following topics:

David Chappell is principal of **Chappell & Associates**, an education and consulting firm in Minneapolis, Minnesota. His clients have included most major computer vendors and many large corporations and government agencies. Earlier in his career, David was a senior software engineer at Cray Research, chaired a U.S. national standardization group, and worked as a pianist accompanying ballet and modern dance classes. More recently, David has been a keynote speaker at conferences in the United States and Europe, appeared frequently on the Computer Channel, and presented seminars around the world. He holds a B.S. degree in economics and an M.S. degree in computer science, both from the University of Wisconsin-Madison. he can be contacted at www.chappellassoc.com.

An important note: this discussion is based on a pre-release version of the ActiveX Software Development Kit (SDK). It's possible that some parts will change before these technologies are finalized. Be aware that what's described here might not exactly match what is finally delivered.

This is the same test you added before the <FORM> tag.

1. What is the ACID test?

{ewc mvimg, mvimage,!answer.bmp}
A. Properties that a transaction should have so it does not fail.

{ewc mvimg, mvimage,!answer.bmp}
B. A test that you can perform to determine whether a transaction failed.

{ewc mvimg, mvimage,!answer.bmp}
C. The change in the state of data.

{ewc mvimg, mvimage,!answer.bmp}
D. A test that can be performed on components before they are placed on a Web server.

2. How do you set a Microsoft Transaction Server component to require a transaction?

{ewc mvimg, mvimage,!answer.bmp}
A. Set the transaction property of the component's package to require a transaction.

{ewc mvimg.mvimage,!answer.bmp}

B. Set the transaction property of the component to support a transaction.

{ewc mvimg.mvimage,!answer.bmp}

C. Set the transaction property of the component to require a transaction.

{ewc mvimg.mvimage,!answer.bmp}

D. Write the component so that it calls the transaction functions **BeginTrans**, **EndTrans**, and **Rollback**.

3. How do you enable an ActiveX server component to participate in transactions on Microsoft Transaction Server?

{ewc mvimg.mvimage,!answer.bmp}

A. Call the transaction functions **BeginTrans**, **CommitTrans**, and **Rollback**.

{ewc mvimg.mvimage,!answer.bmp}

B. Modify the component to call the methods **SetComplete** and **SetAbort**.

{ew

C. Place the component in the Microsoft Transaction Server Explorer.

c mvimg, mvimage,!answer.bmp}

{ewc mvimg, mvimage,!answer.bmp} D. Create the public methods **SetComplete** and **SetAbort** for the component.

4. What are the advantages of stateless objects?

{ewc mvimg, mvimage,!answer.bmp} A. They reduce the server load.

{ewc mvimg, mvimage,!answer.bmp} B. They ensure transaction isolation and database consistency.

{ewc mvimg, mvimage,!answer.bmp} C. They reduce the internal data dependencies of objects.

{ewc mvi D. All of the above.

{mg.mvimage,!answer.bmp}

5. How do you get a reference to a Context object created by Microsoft Transaction Server for a component?

{ewcmvimg.mvimage,!answer.bmp}

A. Call the **GetObjectContext** API.

{ewcmvimg.mvimage,!answer.bmp}

B. Call **CreateObject** ("object.context").

{ewcmvimg.mvimage,!answer.bmp}

C. Microsoft Transaction Server will automatically pass the parameter that gets the **Context** object.

{ewcmvimg.mvimage,!answer.bmp}

D. Call **GetObject** ("object.context").

To complete the lab exercises in this chapter, you must have the required software. For detailed information about the labs and setup for the labs, see Labs in this course.

For background information on this lab, click each of these topics:

Objectives

Prerequisites

Lab Setup

To see a demonstration of the lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

Estimated time to complete this lab: **20 minutes**

Exercises

The following exercises provide practice working with the concepts and techniques covered in Chapter 10: Controlling Access to a Web Site.

Exercise 1: Turning Off Anonymous Logon

In this exercise, you will turn off Anonymous logon to the State University Web site.

Exercise 2: Setting Permissions on a File

In this exercise, you will set permissions on the transcript.asp file to prevent anonymous users from viewing the page.

After completing this lab, you will be able to:

® Turn off Anonymous logon for a Web server to allow only valid Windows NT accounts access to a Web site.

® Set permissions on a Web page to allow only valid Windows NT accounts access to that page.

Before working on this lab, you should be able to:

® Set permissions on files in Windows NT.

® View user account information in Windows NT.

To complete this lab, you need the following:

® Microsoft Windows NT installed with an NTFS partition

® Microsoft Internet Information Server

® The StateU Web site

In this exercise, you will disable Anonymous logon for your Web server so users of the State University Web site will be required to supply valid NT logon names and passwords.

Turn off Anonymous logon

1. Run the Internet Service Manager.
2. Right-click the computer name for the WWW service, and click **Service Properties**.
3. On the **Service** tab, clear the **Allow Anonymous** check box in the Password Authentication group, check the **Basic** check box, and click OK.

   For information about disabling Anonymous logon, see Preventing Anonymous Logon in this chapter.

Test

1. In Microsoft Internet Explorer, go to the home page (default.htm) of the State University Web site.
2. When prompted, enter a valid Windows NT user name and password.

1. If you have disabled Anonymous logon on your Web server, who can access your Web site?

{ewc mvimg, mvimage,!answer.bmp} A. Users with accounts that you have added to a database on your Web server.

{ewc mvimg, mvimage,!answer.bmp} B. Users with valid Windows NT accounts.

{ewc mvimg, mvimage,!answer.bmp} C. Users who log on with the account IUSR_*computername*.

{ewc mvimg, mvimage,!answer.bmp} D. No one can access the Web site.

2. How can you limit database access to users of your Web site?

{ewc mvimg, mvimage,!answer.bmp} A. Create a logon ID for your database that has limited rights, and use that ID to access the database.

{ewc mvimg.mvimage,!answer.bmp}

B. Add records in a new SQL database for each user who needs access to the Windows NT Server and SQL Server.

{ewc mvimg.mvimage,!answer.bmp}

C. Hard code the SQL system administration account and password into the ActiveX Data Objects (ADO) connection string.

{ewc mvimg.mvimage,!answer.bmp}

D. Disable the Internet Guest account and the Guest account.

3. If you have enabled Anonymous, Basic, and NT Challenge/Response authentication for your Web server, what authentication process occurs when a user requests a Web page?

{ewc mvimg.mvimage,!answer.bmp}

A. The Web browser tries Anonymous authentication. If that fails, the user will not be able to access the page and receives an error message.

{ewc mvimg.mvimage,!answer.bmp}

B. If the browser supports NT Challenge/Response authentication, the browser tries to use it. Otherwise, the browser uses Basic authentication.

{ew

C. If the browser supports Basic authentication, the browser uses it.

{ewc mvimg.mvimage.!answer.bmp}

Otherwise, the browser uses NT Challenge/Response authentication.

{ewc mvimg.mvimage.!answer.bmp}

D. The Web browser tries to use Anonymous authentication, but if that fails, the user must provide a valid logon ID and password. Then, if the browser supports NT Challenge/Response authentication, that method is used. Otherwise, the browser uses Basic authentication.

4. How do you configure your Web server to require digital certificates from users who request pages from your Web site?

{ewc mvimg.mvimage.!answer.bmp}

A. Configure the virtual directory of your Web site to use the Secure Sockets Layer (SSL).

{ewc mvimg.mvimage.!answer.bmp}

B. Configure your Web server to use the Secure Sockets Layer (SSL).

{ewc mvimg.mvimage.!answer.bmp}

C. Disable Anonymous logons for your Web server.

{ewc

D. Configure the Active Server Page **Session** object to use digital certificates instead of cookies.

{mvimg,mvimage,!answer.bmp}

5. If you want to restrict access to the page private.asp on your Web site, but also enable all users to access other pages on your Web site, what should you do?

{ewc mvimg,mvimage,!answer.bmp}

A. Configure the hard drive partition, where the Web site files are located, to use the FAT file system.

{ewc mvimg,mvimage,!answer.bmp}

B. Enable Anonymous and Basic authentication for the Web server, and then remove the Internet Guest Account from the access control list for the file private.asp.

{ewc mvimg,mvimage,!answer.bmp}

C. Enable the Guest account, and then remove the Guest account from the access control list for the file private.asp.

{ewc mvimg,mvimage,!answer.bmp}

D. Set the HIDDEN attribute of the file private.asp.

6. Which server-side script retrieves the name of the logon account that is used to access the Web site?

{ewc mvimg, mvimage,! answer .bmp}

A. Request.Server("LOGON_USER")

{ewc mvimg, mvimage,! answer .bmp}

B. Request.ClientCertificates("LOGON_USER")

{ewc mvimg, mvimage,! answer .bmp}

C. Request.ServerVariables("LOGON_USER")

{ewc mvimg, mvimage,! answer .bmp}

D. Session("LOGON_USER")

This article discusses several advanced data access topics, including the types and advantages of cursors, the use of asynchronous queries, and the handling of multiple result sets.

It is intended for advanced programmers with a solid knowledge of both Microsoft Visual Basic programming and data access mechanisms in a client/server environment, with specific focus on Microsoft SQL Server as the database management system of choice.

This white paper includes the following topics:

The complete title of this white paper is *Microsoft SQL Server: An Overview of Transaction Processing Concepts and the MS DTC.*

This paper includes the following topics:

Abstract

The Microsoft SQL Server and BackOffice products running on Windows NT Server have evolved to support huge databases and applications. This paper outlines the history and current power of Microsoft SQL Server, showing that it scales down to small 1-megabyte (MB) personal databases and scales up to giant 100-gigabyte (GB) databases used by thousands of people. SQL Server achieves this scalability today by supporting symmetric multiprocessing (SMP). In the future, Microsoft SQL Server will scale out by partitioning a huge database into a cluster of servers, each storing part of the whole database, and each doing a part of the work. This clustering approach dovetails with the Windows NT Server cluster architecture. Today, Windows NT Server and Microsoft SQL Server clusters provide high availability — but we are still in the early stages of this product evolution. Microsoft intends to extend its cluster architecture to accommodate modular growth, as well as to automate configuration, maintenance, and programming.

This white paper includes the following topics:

Abstract

As thousands of companies rush to establish a presence on the Internet, many are discovering its architecture has potential as an internal corporate network as well as an external publication platform. One industry analyst calls the Internet "the greatest catalyst for change and growth in the IT industry since the PC and LAN 15 years ago."

This paper examines how Microsoft SQL Server enables organizations of all sizes to exploit the power of the Internet. Integrating SQL Server with an Internet or intranet allows organizations to build active Web sites that publish real-time information, provide interaction and customization, conduct business on the Internet securely and reliably, and develop corporate intranets, giving users new tools to access business information, without compromising security and data integrity.

This white paper includes the following topics:

® Introduction

® Why Do I Need a Database on the Active Internet?

® Microsoft SQL Server and the Active Internet

® Optimizing Microsoft SQL Server for the Active Internet

® A Look Ahead

The conventions defined by OLE allow a user to edit an embedded document in place, much as if it were opened in a separate application. With an embedded Microsoft Excel spreadsheet like the one shown in earlier chapters, for instance, the user can activate the embedded object and have access to Excel's commands. Useful as this is, however, an ordinary embedded document doesn't suffice in every situation. Typically, for example, an in-place active document is relegated to whatever area on the screen its container is willing to allot, an area that's usually fairly small. In some cases, the user might want to have the embedded document completely take over the editing area of the user interface. Similarly, when a user prints an ordinary compound document, only the cached presentation appears for any embedded elements — the embedding server's own print functions can't be used. Having a way to access these functions and a few other extra features would let the user see the full functionality of the embedded application rather than just the (admittedly quite large) subset provided by OLE embedding and in-place activation.

The Office Binder, a tool included with Microsoft Office 95, provides a good example of how this can be useful. The idea behind the Binder program is that a user might want to work in a unified way with information created by several Office applications. For instance, imagine a current sales report containing text created with Microsoft Word, quarterly financial data in a Microsoft Excel spreadsheet, and a sales presentation created with Microsoft Power-Point. To collect these disparate kinds of information in a coherent whole, a user might embed the Excel spreadsheet and the Power-Point presentation in the Word document. Another solution would be to embed all three in yet another document — in this case, in a binder.

Figure 11-1 shows an example of the three kinds of data just described embedded in a binder document. As shown in the figure, the binder presents a two-part user interface. On the left appears an icon for each embedded document. On the right is the active document, the Excel spreadsheet. Each of the three documents in this binder is embedded, and the Excel spreadsheet is currently in-place active.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-1. A binder document with three embedded ActiveX documents.

This binder document and the applications that have embedded data within it interact using conventions defined by the *ActiveX Documents* technology. The binder is an ActiveX Documents container, while Excel, Word, and PowerPoint are all ActiveX Documents servers. Each application acts like an ordinary OLE embedded document server, although each one also has a little more functionality. For instance, any ActiveX Documents server is able to take over the entire editing area provided by the container. The container, which in this case is the binder document, essentially gets out of the way and lets the ActiveX Documents server completely control what the user sees. In Figure 11-1, for example, the user can have Excel take over the entire editing area by removing the window on the left containing the icons. An ActiveX Documents server presents a user interface that's more complete than the interface of an ordinary embedded document. To the user, in fact, it looks as if Excel is running independently — the limitations imposed on an in-place active embedded document are gone. Excel really is functioning as an embedded document here, as described in Chapter 8, but it can also offer extra features made possible by the ActiveX Documents technology. (And if you're starting to wonder what all this has to do with the Internet and the Web, be patient — it turns out to be very important.)

When Microsoft introduced the *ActiveX* designation, some technologies formerly assigned the *OLE* label were renamed. OLE Controls, for example, became ActiveX Controls. It's tempting to also assume that OLE Documents became ActiveX Documents, but this is not correct. The former OLE Documents technology is now referred to simply as *OLE*. ActiveX Documents describes a technology that builds upon this older technology — it's not just a new name.

This section contains the following topics:

When Microsoft introduced the *ActiveX* designation, some technologies formerly assigned the *OLE* label were renamed. OLE Controls, for example, became ActiveX Controls. It's tempting to also assume that OLE Documents became ActiveX Documents, but this is not correct. The former OLE Documents technology is now referred to simply as *OLE*. ActiveX Documents describes a technology that builds upon this older technology — it's not just a new name.

All of the things required to present this richer user interface for an embedded document — the ability to take over the container's entire editing window, access to the server's print functions, and so on — are simply extensions to the current embedding and in-place activation features of OLE. Accordingly, containers and servers must first implement embedding and in-place activation and then add a few more interfaces whose methods support the new features. The ActiveX Documents specification defines these extra interfaces.

This section includes the following topics:

To qualify as an ActiveX Documents container, an application must support all the interfaces OLE requires for embedding and in-place activation. As shown in Figure 11-2, ActiveX Documents containers must also support the IOleDocumentSite interface. This interface is implemented on a document site object, the ActiveX Documents analog of the client site object in an OLE container. An ActiveX Documents container provides one instance of a document site object for each embedded ActiveX document. A container can also support the IOleCommandTarget and IContinueCallback interfaces, both of which are discussed later in the Commands topic.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-2. An ActiveX Documents container must implement at least one extra interface in addition to those required by OLE.

As shown in Figure 11-3 on the following page, acting as an ActiveX Documents server requires support for all the server-side embedding and in-place activation interfaces described in Chapter 8 and more. A server might optionally support IPrint and IOleCommandTarget (discussed later), and it must support IOleDocument and IOleDocumentView. Understanding what these two mandatory interfaces do requires first understanding what the word *view* means in this context.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-3. An ActiveX Documents server must implement at least two extra interfaces.

An application such as Word or PowerPoint knows how to manipulate a certain kind of data. Word, for instance, works primarily with text, whereas PowerPoint works with slides and their contents. In each case, the application can present different views of its data. In Word, a user can see a document in Normal view, Page Layout view, or Outline view. PowerPoint allows the user to work with a presentation in Slide view, Outline view, Notes Pages view, and so on. Each view acts like a filter through which the user sees the application's data, each showing the same information in a different way.

In an ActiveX Documents server, each view is represented by a view sub-object. This sub-object must implement the IOleDocumentView interface and might also implement IPrint and/or IOleCommandTarget. The server must also implement IOleDocument, which can be used to create view sub-objects. The container in turn implements one view object supporting IOleInPlaceSite (and perhaps IContinueCallback) for each view sub-object in the server.

Figure 11-2 includes one other interface, called IOleContainer, which allows a server to enumerate the objects managed by its container. Although this interface is not strictly required for an OLE container, IOleContainer turns out to be quite useful and so is commonly supported in the situations discussed in this chapter.

When a user prints a document directly from Word, what is actually printed depends on the view Word is currently displaying. If the user is looking at the document in Outline view, for example, the document's outline is printed. When a user prints a document from Word acting as an ActiveX Documents server, the same thing should occur. To allow this, a view sub-object can support IPrint. Using this interface, an ActiveX Documents container can ask a particular view sub-object in the server to print its view of the data. No longer does printing an embedded document mean that only the document's cached presentation is printed; with ActiveX Documents, the server itself can control exactly what is printed.

Printing can be a lengthy process, and users might get bored or change their minds about the wisdom of their print request. Once an ActiveX Documents container has asked a server to print a document, that server should periodically call the FContinuePrinting method in its container's IContinueCallback interface. This method's parameters include the number of the page currently being printed and the number of pages printed so far. A container might use these to keep its user apprised of the server's progress in printing. If the user tells the container to cancel the print job, the container can pass this information on to the server by setting an appropriate return code on FContinuePrinting. When the call returns to the server, it checks this code and, if necessary, cancels the print job.

Both a container and a server can support the IOleCommandTarget interface. It's easiest to think of this interface as a stripped-down version of IDispatch. Recall that a dispinterface assigns DISPIDs to a group of methods and then lets a client invoke any method in that dispinterface using the single vtable method IDispatch::Invoke. The dispinterface itself is assigned a GUID, allowing the same DISPIDs to be used in different dispinterfaces without fear of ambiguity. With IOleCommandTarget, various *command groups* can be defined, each of which is assigned a GUID. Each command in a command group is assigned an integer value, analogous to the DISPIDs in a dispinterface. To execute any command, a client of IOleCommandTarget can invoke the Exec method of IOleCommandTarget, providing the GUID that identifies a command group along with an integer identifying a command in that group. It's also possible to pass a command with parameters using variants, the same mechanism used by IDispatch.

Why invent a new interface when IDispatch would certainly have sufficed? The answer is that the creators of ActiveX Documents felt that IDispatch was too heavyweight for the simple requirements here. The primary reason for using commands at all in this context is to allow a container to ask a server to perform such tasks as displaying its properties and to ensure that toolbar commands work as expected. Accordingly, an ActiveX Documents container and server typically exchange straightforward commands such as Open, Save, and Copy. Using IDispatch for such simple operations was seen as needlessly complex.

Because the interfaces required for using ActiveX Documents are simply extensions of those already used for OLE embedding and in-place activation, the interactions between an ActiveX Documents container and server are very similar to those between an OLE container and server. As in OLE, an ActiveX Documents container (such as a binder document) loads an appropriate server (such as Excel or Word). The container then initializes the server using one of the IPersist* interfaces. A binder stores all its embedded documents' data in a single compound file, each in its own storage. This isn't the only choice, however. An ActiveX Documents container can also initialize a server from a file or from some other persistent storage, assuming that the server supports the appropriate IPersist* interface.

As part of the ordinary initialization process for an embedded document, a container invokes a server's IOleObject::SetClientSite method. In OLE, the container passes a pointer to its IOleClientSite interface as a parameter on this method. In ActiveX Documents, however, the container passes a pointer to its IOleDocumentSite interface instead. From this, the server can determine whether its container views it as an ordinary OLE embedded object or as an ActiveX Documents object. When the user requests that an ActiveX document be activated by, say, double-clicking on it, an ActiveX Documents container invokes the server's IOleObject::DoVerb method as always. An ActiveX Documents server responds to this differently than an ordinary OLE embedding server does, however. When it receives a call to this method, the ActiveX Documents server invokes the only method in its container's IOleDocumentSite interface, the whimsically named ActivateMe, to request that its container make it active. The container can respond by using QueryInterface to ask the server for a pointer to its IOleDocument interface. The container then invokes IOleDocument::CreateView, which creates a new view sub-object in the server and returns a pointer to that object's IOleDocumentView interface. Using this interface's methods, the container can activate the view, work with it, and close it when it's no longer needed.

What does all this have to do with the Internet or the Web? Well, initially, nothing at all. ActiveX Documents objects were originally known as *Document Objects*, or just *DocObjects*, and they were first widely disseminated in the Office Binder program. The Binder is a useful tool, but it was created with a desktop-centric focus. What DocObjects provided, though, turned out to be useful in a much broader context. By supporting only a few extra interfaces in addition to those already required by OLE, one application could host another while still allowing a user to access the complete range of the hosted application's features. Those few extra interfaces brought with them the ability to work with everything the embedded application had to offer. The user could see a common frame yet work naturally within that frame with all types of data.

A binder document is one example of a common frame through which a user can access different applications. Another example, one that's much more interesting today, is a web browser. It too can provide a common frame for accessing and working with all kinds of data and all kinds of applications. It was this realization — the tie to the Web — that prompted the name change from Document Objects to ActiveX Documents. As described next, the result was a complete revamping of Microsoft's web browser and ultimately of the Windows user interface itself.

When the HTML viewer loads a document, that document might contain one or more embedded scripts. Those scripts can make use of the programmable objects exposed by the viewer, along with any objects that are loaded dynamically. Today the two leading languages for writing scripts embedded in HTML are Netscape's JavaScript and Microsoft's Visual Basic Script (formally known as Visual Basic Scripting Edition but commonly called VBScript). Java-Script is syntactically similar to the Java programming language, whereas VBScript is a subset of Visual Basic. It's not hard to imagine that other languages might be used for scripting as well.

The HTML viewer itself has no reason to either know or care what language an executing script is written in. The script executes in a separate component called a *scripting engine,* while the viewer acts as a generic host for this engine. The viewer can instantiate a scripting engine, hand it a script, and tell it to begin executing the script. As the script executes, it can invoke methods in the viewer's objects and receive events from those objects. The interfaces supported by the HTML viewer and a scripting engine that make all this possible are defined by the *ActiveX Scripting* specification. (ActiveX Scripting was originally known as *OLE Scripting*.)

Furthermore, because the HTML viewer can also act as an ActiveX control container, loading an HTML page can result in loading one or more ActiveX controls as well as a script. Scripts executed by a scripting engine can interact not only with the built-in objects in the HTML viewer but also with any loaded controls. (In fact, an executing script can't distinguish between the two.) The relationships among the HTML viewer (acting as an ActiveX Scripting host and an ActiveX control container), a scripting engine, and a pair of ActiveX controls are shown in Figure 11-7. And finally, although this discussion uses only the HTML viewer as an example of an ActiveX Scripting host, this technology is in no way specific to this application. Any application can become an ActiveX Scripting host and then load and be driven by any scripting engine.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-7. The HTML viewer is both a host for ActiveX scripting engines and a container for ActiveX controls.

This section includes the following topics:

A scripting engine is a COM object, generally implemented as an in-process server, that is capable of executing a set of scripts — for instance, all those written in a particular language. Internet Explorer 3.0, for example, includes scripting engines for both VBScript and JavaScript. An ActiveX Scripting host typically implements objects whose methods, properties, and events can be invoked, accessed, and received by an executing script. For the HTML viewer, these objects include the Window object and the Document object, described in the previous section. The host can load objects such as ActiveX controls dynamically as well.

Figure 11-8 illustrates the objects and interfaces that can be implemented by an ActiveX Scripting host. As the figure shows, a host implements a *scripting site object* that supports the IActiveScriptSite interface. Using the methods in this interface, a scripting engine can acquire pointers to the interfaces of top-level objects the host makes available, inform the host of errors that occur, notify the host that the script has completed, and more. If the object supporting IActiveScriptSite provides its own user interface, it can also support IActiveScriptSiteWindow, allowing a scripting engine access to that object's window. Each object in the host, such as the Window and Document objects in the HTML viewer or a loaded ActiveX control, implements its own IDispatch interface, allowing a scripting engine to invoke its methods and access its properties. Each object should also implement IProvideClassInfo (or perhaps IProvideClassInfo2), allowing its client to access its type information. And finally, host objects that generate events also implement IConnectionPoint and IConnectionPointContainer.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-8. The interfaces that an ActiveX Scripting host and its objects can implement.

Figure 11-9 illustrates the interfaces that a scripting engine can support. Every scripting engine must support the IActiveScript interface. A host uses the methods in IActiveScript to pass the scripting engine a pointer to the host's IActiveScriptSite interface, to tell the script to begin executing, and to perform other tasks. If the scripting engine can load scripts from persistent storage, it also supports one or more of the IPersist interfaces, such as IPersistStorage, IPersistStreamInit, or IPersistPropertyBag. Scripting engines that allow script text to be added dynamically can support IActiveScriptParse, which lets a host such as the HTML viewer pass in a script received as part of an HTML file. If an error occurs during execution of a script, the engine passes its host a pointer to the IActiveScriptError interface, which is implemented by a distinct object in the engine. By calling methods in this interface, the host can learn more about the error. Finally, scripting engines that can accept events sent by a host or that allow a host to access the script's methods and properties must also implement IDispatch.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-9. The interfaces that an ActiveX scripting engine can implement.

To understand how all this works, imagine that Internet Explorer 3.0's HTML viewer, an ActiveX Scripting host, loads the following very simple HTML document:

```
<HTML>
<TITLE>ActiveX Scripting Example</TITLE>
<BODY>
<H1>Illustrating Scripting</H1>
<SCRIPT LANGUAGE=VBScript>
  document.bgColor = "White"
  document.write "<HR>"
  document.write
   "Hello from the VBScript scripting engine"
  document.write "<HR>"
</SCRIPT>
</BODY>
</HTML>
```

When the HTML viewer loads this document, it happily reads and interprets the first few lines using the HTML tags in the angle brackets. For example, the IE 3.0 viewer renders the line <*H1*>*Illustrating Scripting*</*H1*> as a level-one heading (based on the *H1* tag) as shown in Figure 11-10. When the viewer encounters the next line, however, beginning with the *SCRIPT* tag, it knows that it will need to load a scripting engine. Examining the *LANGUAGE* parameter, it determines that a VBScript engine is required. (If this were a JavaScript example, the value of the *LANGUAGE* parameter would be *JavaScript*.) The HTML viewer looks up *VBScript* in the registry — it's a ProgID, which is described in Chapter 4 — and finds the associated CLSID. The viewer then calls CoCreateInstance with this CLSID to create an instance of the VBScript scripting engine and get an initial pointer to it.

Once the engine is running, the host can acquire a pointer to the engine's IActiveScript interface. The host loads the HTML file's script into the scripting engine using methods in IActiveScriptParse and then invokes the scripting engine's IActiveScript::SetScriptSite method, providing a pointer to its own IActiveScriptSite interface. The basics are now in place for the host and the scripting engine to perform their complementary tasks.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-10. The result of loading the example HTML file.

The script that the engine is executing will need to use one or more of the objects supported by the host. Our VBScript example, for instance, sets the bgColor property and invokes the Write method of the HTML viewer's Document object. (Figure 11-10 shows the results.) To set properties and invoke methods of an object in its host, the scripting engine needs a pointer to that object's IDispatch interface. To allow the engine to get this pointer, the host can call IActiveScript::AddNamedItem for one or more of its objects, passing in the object's name as a character string. The HTML viewer, for instance, makes this call for the Window object, the top-level object in its hierarchy. This call isn't necessary for the lower-level objects in the hierarchy, however. Instead, the scripting engine can acquire pointers to these objects through properties on the Window object, as explained earlier.

Once the host has informed the scripting engine about all the necessary objects, the host invokes IActiveScript::SetScriptState to tell the engine to begin executing its script. When the scripting engine needs a pointer to an object that it learned about through IActiveScript::AddNamedItem, it calls IActiveScriptSite::GetItemInfo with the name of that object. In our example, the scripting engine calls this method only once, requesting information about the Window object. This call returns a pointer to the IUnknown interface of the named object. The scripting engine calls QueryInterface on the returned IUnknown pointer, asking for IDispatch. When it acquires the Window object's IDispatch pointer, the scripting engine next uses it to access this object's Document property and acquires a reference to the subordinate Document object. Using this reference, the scripting engine can set the Document object's bgColor property and invoke its Write method, as specified in the script.

A scripting engine calls its host to invoke methods and access properties. But a host might need to call a scripting engine, too, to inform it of events that have occurred. If an object in the host displays a button, for instance, the object might need to inform the scripting engine that the user has clicked on the button and that some event-handling code in the script should run. This is not a new problem — the process is just like an ActiveX control sending events to its container. Happily, the solution adopted by ActiveX Scripting is identical to that defined for ActiveX Controls. (In fact, the object sending the events to the script might actually be an ActiveX control.) Hosts such as the HTML viewer can provide type libraries for their objects, just as ActiveX controls do. A scripting engine gets a pointer to an object's type library and then reads the type library to learn how to build

sinks for that object's events. This process is very similar to what control containers do (described in Chapter 9). And, as with controls, connection points are used to pass the necessary pointers from the engine to the objects to allow the events to be sent and received.

By standardizing the interactions between an executing script and the objects it uses, ActiveX Scripting allows any host to work with any scripting engine. If the simple script shown earlier were written in JavaScript rather than VBScript, for example, nothing would change from the host's point of view, except that it would instantiate a different class of scripting engine. It's even possible to mix VBScript, JavaScript, and (potentially) other scripting languages in the same HTML file and have each script executed by its own scripting engine. And, as mentioned earlier, ActiveX Scripting is useful for more than scripts loaded into a browser — scripting capabilities can be added to any application by implementing the interfaces required of an ActiveX Scripting host.

Visual Basic was the first widely used container for ActiveX controls, and its requirements were a driving factor in their original design. However, Internet Explorer 3.0's HTML viewer is a control container, too. An HTML page might contain data, for instance, that requires loading a specialized ActiveX control into the client's browser to view it. The code for this control might already be present on the client machine, or it might be downloaded from a web server when it's needed. Alternatively, IE 3.0 might download a platform-independent *applet* written in the Java language. (An applet is a program, typically a fairly small one, that runs inside a container of some kind, such as a web browser.) In either case, the result is the same: code is loaded as needed (perhaps from a web server) and executed on the browser's machine.

While the most visible effect of the Internet's collision with controls is probably their current name — ActiveX controls rather than OLE controls — the emergence of web browsers as control containers also caused some of the original, desktop-centric design decisions concerning controls to be revisited. As Chapter 9 explained, for example, the requirements a COM object must meet to qualify as an ActiveX control have been greatly reduced, a change made largely to accommodate the process of loading controls over slow Internet links. Loading potentially large amounts of data over those slow links has also led to extensions to the original controls technology. This section examines how ActiveX controls interact with browsers and discusses how a control can better fit into this new environment. Although Internet Explorer 3.0 serves as the example throughout, all the HTML shown here conforms to standards set by the World Wide Web Consortium — it contains nothing specific to Microsoft's browser.

This section includes the following topics:

Using the *OBJECT* tag in HTML, IE 3.0's HTML viewer can load and use any ActiveX control. And just as scripts can be written that use the objects built into the HTML viewer, so too can scripts make use of dynamically loaded controls. For example, suppose that the HTML viewer loads the following page:

```
<HTML>
<TITLE>HTML Control Example</TITLE>
<BODY>
<H1>Click An Arrow</H1>
<P>
<OBJECT
  CLASSID="clsid:B16553C0-06DB-101B-85B2-0000C009BE81"
  ID=SpinButton
  HEIGHT=200
  WIDTH=100
  HSPACE=85
>
</OBJECT>
<SCRIPT LANGUAGE=VBScript>
Sub SpinButton_SpinUp()
  MsgBox "(Up arrow clicked)"
End Sub
Sub SpinButton_SpinDown()
  MsgBox "(Down arrow clicked)"
End Sub
</SCRIPT>
</BODY>
</HTML>
```

After the heading *Click An Arrow*, this document uses HTML's *OBJECT* tag to load an ActiveX control — in this case, it's the spin button control you saw in Chapter 9 ("An Application Developer's View," page 210). When this document is loaded, the HTML viewer reads the CLASSID attribute and then calls CoCreateInstance with that CLSID. The ID attribute gives the control a name that can be referred to in the script, while the remaining attributes determine the control's size and position on the page. There's nothing special about the control — this is the same code that was loaded into Visual Basic in Chapter 9's example. As in that example, the code is loaded locally, not from a web server. If no COM object is available locally with this CLSID, this example won't work.

Following the *OBJECT* tag is a simple VBScript program that again emulates the example in Chapter 9. Recall that the spin button control generates events when a user clicks on either of its arrows, events that can be caught when the control is loaded into Visual Basic. A VBScript program can also catch those events when the spin button control is loaded into the HTML viewer. (The previous section on ActiveX Scripting explained how the VBScript scripting engine is able to receive those events.) As the example shows, simple subroutines similar to those shown in Chapter 9 can be written in VBScript and executed when the user clicks on an arrow and the control generates an event. Figure 11-11 shows what a user sees after clicking on the up arrow.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-11. The result of loading the example HTML file and then clicking on the up arrow.

An ActiveX control usually has persistent data that it must load when it begins executing. Controls loaded from HTML files have several options for where this persistent data is kept and how it is loaded. This section describes these choices.

Loading Small Amounts of Data

How a control loads its persistent data depends on what kind of persistent data it has. Suppose, for instance, that a control has several properties whose values need to be set when the control is loaded. As shown here, those values can be stored in the HTML file itself using the *OBJECT* tag's *PARAM* element:

```
<OBJECT
  CLASSID="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
  ID=label1
  WIDTH=150
  HEIGHT=500
>
<PARAM NAME="Angle" VALUE="270">
<PARAM NAME="Alignment" VALUE="2">
<PARAM NAME="Style" VALUE="0">
</OBJECT>
```

When IE 3.0's HTML viewer encounters this *OBJECT* tag, it loads the code for the specified control (which we'll again assume is already present locally) and requests a pointer to that control's IPersistPropertyBag interface. The viewer then reads the *PARAM* elements and hands their values to the control one at a time, as described in Chapter 5. (See "The IPersistPropertyBag Interface," page 125.)

This approach works well with controls that can reasonably store their properties as text in an HTML file. But other controls might store their persistent data in a binary form and expect to load this data through IPersistStream. To allow this, the *OBJECT* tag can use the DATA attribute to specify a file that contains the data:

```
<OBJECT
  CLASSID="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
  ID=chart1
  WIDTH=200
  HEIGHT=500
  DATA="http://www.acme.com/charts/profits.ods"
  >
</OBJECT>
```

When Internet Explorer's HTML viewer encounters the DATA attribute, it fetches the indicated file and hands it to the control as a stream through IPersistStream::Load. Although it's not shown here, it is also possible to place a limited amount of data for a control directly in the HTML file's DATA attribute.

Loading Large Amounts of Data

Both examples shown so far work well with controls that have a relatively small amount of persistent data. But imagine a control whose persistent data includes large graphic or video files or other *binary large objects* (BLOBs). In this case, the control's BLOB data is certainly too big to be stored in the HTML file. It might also be impractical to store this data in the file named with the DATA attribute. Because the file would be loaded using IPersistStream, the file is handed to the control as a complete unit and all data in the file must be present on the local machine before the control can see any of it. Preventing the control from becoming even partially active until all the data has arrived is a less than optimal solution when that data is being loaded over a slow Internet link — users get frustrated when they're forced to spend much time looking at an hourglass icon.

A better approach would be to initialize the control with all its "small" persistent data and then load any BLOBs asynchronously. Web browsers do this today with ordinary HTML pages, first loading the page's text and then fetching any embedded images. The benefit is that the user sees an active (although incomplete) page almost immediately and gradually gets the larger data elements which complete the page. Controls with BLOB data can work the same way — first loading any smaller data and becoming at least partially active to the user before gradually loading BLOB data.

This two-part initialization scheme relies on *data path properties*. A data path property is like any other property a control might support except that its value can be a URL. Data path properties are stored in the file identified by the DATA attribute of the *OBJECT* tag and are passed to the control through IPersistStream. When the control receives its properties, it examines them individually and uses their values to initialize itself. When the control recognizes a data path property, however, it can extract the property's URL and use it to locate and load the data it refers to.

The URL contained in the data path property can be absolute, containing everything needed to locate the machine on which the data resides. For example, the data path property shown in Figure 11-12 contains an absolute URL. More likely, however, a data path property's value is a relative URL, which must be combined with a base URL (such as that of the page in which the control is embedded) to completely specify the data's location. Because only the control's container knows this base URL, the container is typically involved in the process of locating the data identified by a data path property. To allow this involvement, the container implements the IBindHost interface.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-12. Three properties for a control, one of which is a data path property.

When a control needs to load information identified by a data path property, it can invoke its container's IBindHost::CreateMoniker method, passing in the URL contained in the data path property. The host creates a moniker (such as a URL moniker) that identifies the absolute location of the data and returns a pointer to that moniker back to the control. The control is then free to call the moniker's IMoniker::BindToStorage method to retrieve the information referred to by the data path property. Normally, however, a well-behaved control won't do this. Instead, it allows its container to participate in the binding process. The container, for instance, might have loaded several controls, each containing data path properties referencing remote BLOBs and all loading those BLOBs at the same time. The container might need to prioritize the order in which BLOBs are loaded, based on information only it knows.

Accordingly, rather than calling IMoniker::BindToStorage directly, a control typically calls its container's IBindHost::MonikerBindToStorage method, as shown in Figure 11-13, passing a pointer to the moniker received from the container. The container then calls this moniker's BindToStorage method. If the moniker in question is a URL moniker, as it usually is, the information referenced by the data path property (the control's BLOB) is now downloaded asynchronously into a stream provided by and accessible to the control. The URL moniker keeps the control informed of the arrival of new chunks of data by periodically invoking OnDataAvailable in the control's implementation of IBindStatusCallback. (The control passes a pointer to this interface as a parameter on MonikerBindToStorage, and the container passes it to the moniker through the bind context object, as described in Chapter 6; see "How Asynchronous Monikers Work," page 148.)

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-13. Moniker binding for a data path property.

The benefit of all this complexity is that a control with BLOB data can become at least partially active quickly and then load larger files in the background a bit at a time. This makes for happier users, who aren't required to wait for all the control's data to arrive before beginning to use that control. And should a control find itself loaded into a container that doesn't support IBindHost, it can attempt to fend for itself by converting its data path properties into monikers using MkParseDisplayNameEx and directly calling BindToStorage on those monikers.

A control with data path properties must take one more action, however. When downloading data using a URL moniker, the control eventually receives an indication from the moniker that all the data has been loaded. The control must then inform its container that initialization has been completed and that it is fully ready for use. To do this, the control can send the OnReadyStateChange event to its container. The control can also set the value of a property called ReadyState, which the container can use to query the control's state. Through this event and/or property, the control can indicate different states: it has loaded all properties except asynchronously loaded BLOBs, it has loaded all properties including BLOBs, and so on.

ActiveX controls such as the spin button control that were created before the advent of these new Internet-related technologies don't take advantage of these new features. Although older controls can be loaded and used by control container web browsers, they don't provide all the benefits of a control written with the Internet in mind. Controls that are Internet-aware are made more efficient with support for data path properties and asynchronous downloading along with the OnReadyStateChange event and/or the ReadyState property. These features are by no means required, but they make a control much better suited for use inside a web browser.

In the examples shown so far, a control's data might have been stored on a remote machine, but the code for the control was assumed to be resident on the browser's machine. This need not be the case, however. Why not load a control's code from a web server when it's needed? To tell the browser where the code is, the *OBJECT* tag can include a CODEBASE attribute. Here's a simple example:

```
<OBJECT
 CLASSID="clsid:B16553A0-06DB-101B-85B4-0000C009BE05"
 CODEBASE="http://www.acme.com/welcome/mapshow.ocx"
 DATA="http://www.acme.com/maps/campus.geo"
 ID=MapDisplay
 HEIGHT=450
 WIDTH=450
>
</OBJECT>
```

When Internet Explorer 3.0 encounters this tag in an HTML file, it downloads the file named by the CODEBASE attribute (assuming that no code for this CLSID is currently present on the machine) and then instantiates the control. In this example, the referenced object is an ActiveX control, but Internet Explorer 3.0 also supports the downloading of Java applets. An attribute called CODETYPE on the *OBJECT* tag can be used to indicate the MIME (Multipurpose Internet Mail Extensions) type of this object, such as *application/javavm*, which lets the browser decide whether it's worthwhile to download it. And as the example shows, it's also legal to use the CODEBASE and DATA attributes at the same time, causing the browser to download both a control's code and its persistent data.

This section includes the following topics:

MIME types are used throughout the Web environment to indicate data types. Other commonly seen MIME types are *text/html, image/gif, image/jpeg,* and *video/mpeg*. At the time this book is being written, no permanent MIME type has yet been defined for ActiveX controls.

While the length of time required to move from abstract concept to widespread deployment in software hasn't changed radically (writing code still takes time), the interval between development of a new concept and widespread assimilation of that concept certainly has. No technology better demonstrates this change than Java. Created by Sun Microsystems, Java is a programming language, one not too different from C++. But Java is more, too, offering exciting possibilities for the Internet and for COM.

As with most programming languages, it's possible to compile a program written in Java and produce a binary executable. This isn't commonly done today, however. Instead, Java source code is usually translated into a machine-independent *bytecode* rather than a machine-specific binary. This bytecode is then interpreted by the Java Virtual Machine (VM), software running on a real machine. Using this scheme, the same Java code can be executed on any machine that supports the Java VM.

One popular use of Java is to create applets, relatively small Java programs that run inside a container such as a web browser. Since Java applets can be distributed as bytecode rather than as machine-specific binaries, the same applet can be downloaded and executed on different systems. All that's necessary is for the target machine to have Java VM software available. It's also possible to create stand-alone applications in Java. Unlike applets, applications don't assume the existence of a container.

This section includes the following topics:

Microsoft has wholeheartedly endorsed the Java language. Microsoft's Java development tool, Visual J++, allows the creation of both applets and applications. At first glance, it might not be obvious why Microsoft would choose to support this new language so strongly. After all, Java was created by Sun, a direct competitor. Furthermore, the machine-independent nature of Java's bytecode has led many to suggest that this new development tool could weaken the dominance of Windows/Intel systems. Despite this, however, Java offers a benefit that's very attractive: it meshes exceptionally well with COM. Although COM is officially language neutral, it's fair to say that COM and its supporting technologies were designed with C++ and Visual Basic in mind. Remarkably, even though it was created in a completely separate environment by a competing company, Java actually fits with COM as well as or even better than these two languages. A key part of this fit is that Java objects, like COM objects but unlike objects in C++, can support multiple interfaces. This, together with a few other features, makes Java an excellent language with which to implement and use COM objects. While this sort of technical serendipity is more the exception than the rule, Java and COM really are a natural pair.

Microsoft's implementation of the Java Virtual Machine integrates Java objects and COM objects. Part of this integration is that from the point of view of a COM client, the Java VM makes a Java object appear to be just another COM object. With Java applets, for example, Microsoft's Java VM automatically constructs a dispinterface containing all the applet's public methods. With other Java objects, vtable interfaces are created. These methods are then accessible to clients of this object through a VM-provided implementation of IDispatch, as shown in Figure 11-14. To complete the illusion, the VM provides an implementation of IUnknown for each Java object, allowing clients to acquire pointers to other interfaces the object supports. The VM also implements a class factory, allowing a client to treat Java objects like any other COM objects. The Java programmer creates objects as usual — nothing special is required. All the services necessary to make those objects look like COM objects are supplied transparently by Microsoft's Java VM.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-14. Microsoft's Java VM lets a Java applet look like a COM object.

Microsoft's Java VM also provides the reverse translation: from the point of view of a Java object, an external COM object looks exactly like a Java object. Again, this integration is achieved without making any changes to the Java language itself. Instead, the Java VM transparently performs the necessary translations to map between the two kinds of objects.

Java is an excellent tool for creating COM clients, as the Java environment offers services that make life significantly easier for COM programmers. For example, a programmer working with COM objects in C++ must always be aware of reference counting. For a C++ client, this means calling Release whenever an interface pointer will no longer be used. Java programmers need not concern themselves with reference counting, however. Instead, the Java VM notices when an object is no longer referenced and automatically deletes it, a service known as *garbage collection*. When Microsoft's Java VM notices that the "garbage" object being collected is a COM object, it simply calls Release on the object. Unlike C++ COM clients, the creator of a COM client in Java never needs to worry about keeping track of which objects are no longer needed and then releasing them.

For acquiring references to new interfaces on an object, Microsoft's Java VM even hides calls to QueryInterface beneath the Java language's built-in operators. A Java programmer writes the same code to access a new interface regardless of whether that interface is on a Java object or a COM object. (In fact, the Java programmer can't tell them apart.) For a COM object, however, the Java VM intercedes, silently calling QueryInterface on the object and returning the new interface pointer. Unlike C++ developers, COM programmers working in Java never need to make explicit QueryInterface calls.

In order to provide all the translations required to map between Java and COM, Microsoft's implementation relies on the information stored in a COM object's type library. And to further integrate Java into the COM world, Microsoft offers Java class libraries exposing key COM functions such as CoCreateInstance, along with access to monikers, Structured Storage, and more. Although neither Java nor COM was designed with the other in mind, the two fit together very well.

Using Java to create COM objects and to write clients that access COM objects is an appealing idea. By hiding some of the rough edges, Java makes using COM that much easier. But a key purpose of Java, creating downloadable applets that run in web browsers, has no intrinsic connection to COM. How does Internet Explorer 3.0 support this?

Since Microsoft's implementation of the Java Virtual Machine makes a Java object look like a COM object, supporting Java applets is no different than supporting COM objects. The Java VM is implemented as an ActiveX control included with Internet Explorer 3.0. To execute a Java applet, the applet is simply loaded together with this control. To a control container such as Internet Explorer's HTML viewer, the applet looks like any other ActiveX control. And Microsoft's ActiveX control implementation of the Java VM can execute any standard Java applet, not only those created using Microsoft Visual J++.

Implementing the Java VM as an ActiveX control has broader implications, too. Since applets look like ActiveX controls, and since controls can be driven by scripts, Java applets can also be scripted. Using the ActiveX Scripting interfaces, VBScript, Java-Script, or another scripting language can be used to access the methods exposed by an applet. Java applets can also work with other applets and ActiveX controls in the same page. Finally, because the Java VM ActiveX control makes any Java applet look like a control, an applet can be loaded into any ActiveX control container and behave just as if it were a control. Although Java applets have historically relied on web browsers as containers, they can now be used with other control containers as well.

As with ActiveX controls, the *OBJECT* tag can appear in an HTML page to indicate that a Java applet should be downloaded. Internet Explorer 3.0 also supports the *APPLET* tag, an older mechanism for embedding Java applets in HTML pages. When Internet Explorer 3.0 encounters an *APPLET* tag, it internally converts it to an *OBJECT* tag with the CLSID of the Java VM's ActiveX control. Internet Explorer then loads the Java VM ActiveX control and passes it the *APPLET* tag's parameters. The control then does everything required to download and run the applet.

Once downloaded, a Java applet can potentially call other COM objects or native code on the system. Ordinarily, an applet is sand-boxed, as described earlier, and so isn't allowed to make these calls. As with ActiveX controls, however, Internet Explorer 3.0 allows a Java applet to be digitally signed and to have this signature checked when it's downloaded. Assuming that the signature identifies a trusted source, the applet is permitted to call other COM objects and local code just as a trusted ActiveX control would. For example, because any COM object looks like a Java object to an applet, it's possible for a digitally signed applet to access the automation services that many applications provide. A Java applet might access Excel's built-in services, for example, as a Visual Basic program might do.

Part of the reason for the tremendous growth of the World Wide Web is surely the appeal of its fundamental metaphor: browsing. The central notion underlying browsing is the idea of hyperlinks. To a user, a hyperlink appears on the screen as colored or underlined text, or as a graphic element embedded in the page, or perhaps in some other way. Clicking on a hyperlink changes what the user sees. In some cases, clicking on a hyperlink in an HTML document might simply result in displaying another part of that same document. In other situations, clicking on a hyperlink results in loading an entirely new document.

Most users like the browsing paradigm — it's easy to learn and powerful to use — and Microsoft intends to integrate it throughout the Windows and Windows NT user interface. Key to this is finding a way to provide hyperlinks between all kinds of elements, not just HTML documents. Why can't we create a hyperlink between, say, a Word document and an Excel spreadsheet? Rather than embedding or linking the two documents using the conventions of OLE, why not tie them together with a hyperlink as if they were HTML documents? This is the goal of *ActiveX Hyperlinks*. By enabling the creation of hyperlinks that reference all kinds of elements, including but not limited to HTML documents, and by wrapping this generality in standard COM interfaces, ActiveX Hyperlinks applies the browsing metaphor to a broad range of documents and applications.

This section includes the following topics:

An ActiveX hyperlink is a COM object that supports the IHlink interface. It also supports IPersistStream, allowing its persistent state to be saved to and loaded from a stream, and IDataObject, allowing its contents to be copied using drag and drop or the clipboard. Every ActiveX hyperlink object contains (at least) three key pieces of information:

® A friendly name that can be displayed to the user when the hyperlink is visible. (Showing the friendly name is not required, however, because how a hyperlink is displayed is ultimately determined by the container that displays it, not by the hyperlink itself.)

® A moniker for the hyperlink's target — that is, for the application and data to which the hyperlink points.

® A string indicating a specific location within the target.

For example, an ActiveX hyperlink to a Word file on a local machine might contain a friendly name such as *Current Status Report*, a file moniker that references the Word file, and a string indicating a location such as a Word bookmark within that file, as shown in Figure 11-15. An ActiveX hyperlink to an HTML document stored on the Internet might contain a friendly name such as *Acme Product Support Info*, a URL moniker that references the link's HTML document, and a string identifying a location within that document. A developer might use this second hyperlink to add an option to an application's help menu that directly connects the user to product support information on the World Wide Web.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-15. Two example ActiveX hyperlink objects and their contents.

All ActiveX hyperlinks look the same to their clients, who see them primarily through the methods in IHlink. Those methods include the following:

® The **GetFriendlyName** method can be used by a client to learn the friendly name of the ActiveX hyperlink.

® The **GetMonikerReference** method returns the moniker and the location string from the ActiveX hyperlink.

® The **Navigate** method causes the ActiveX hyperlink to navigate to its target, the document to which it points.

To create an ActiveX hyperlink object, a container need only call one of several standard library functions and pass in the appropriate data. HlinkCreateFromMoniker, for example, lets a container create a hyperlink object by providing the three required components of an ActiveX hyperlink: a moniker, a location string, and a friendly name. HlinkCreateFromString lets a container create an ActiveX hyperlink object by providing a location string, a friendly name, and a character string identifying the hyperlink's target.

However it is created, an ActiveX hyperlink object communicates with its container through the container's implementation of IHlinkSite, as shown in Figure 11-16, and communicates with its target through the methods in IHlinkTarget. Note that a hyperlink can refer either to a location in the currently displayed document or to a location in another document. Supporting this first case requires the hyperlink's container to itself implement IHlinkTarget.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-16. ActiveX hyperlinks, their targets, and a container.

Although they aren't shown in the figure, two other components play a part in ActiveX hyperlinking: the *browse context object* and *hyperlink frames*. The browse context object supports the IHlinkBrowseContext interface, and it is responsible for maintaining the *navigation stack*. This data structure supports an integral part of the browsing metaphor: the ability to move forward and back in the list of visited documents. A traditional web browser maintains this list itself, but it applies only to hyperlinks between HTML documents. Because no single "browser" application might be able to encompass all the documents a user visits through ActiveX hyperlinking, an external object must maintain a list of visited documents. The navigation stack maintained by the browse context object generalizes the traditional web browser history list to include all documents browsed using ActiveX hyperlinks, including HTML documents, Word documents, Excel spreadsheets, or anything else.

Finally, navigating to a hyperlink should ultimately result in displaying something new to the user. To provide some consistency, it's common (though not mandatory) to wrap a single frame around a succession of displayed documents accessed with ActiveX hyperlinks. Internet Explorer 3.0, for example, can be used to browse across many different kinds of data, and it gives the user a common frame for all of them. It can be useful to keep this frame informed about what's happening, allowing it to do whatever is needed to maintain a smooth look for the user. For example, all applications hosted within a hyperlink-aware frame can rely on the frame to locate the browse context object for them. To do this, a frame supports IHlinkFrame, whose methods are called by various components in the hyperlinking process at appropriate times. Internet Explorer 3.0's simple frame,

IEXPLORE.EXE, implements this interface, as will Internet Explorer 4.0.

When a user clicks on a hyperlink, the container that receives the click creates an ActiveX hyperlink object containing the correct information and passes it a pointer to its IHlinkSite interface. (With the creation functions mentioned earlier, such as HlinkCreateFromString, all this can be done with a single function call.) Once the hyperlink object is running, the container calls its IHlink::Navigate method. To find out whether this hyperlink refers to another location in the document the container is currently displaying or to a location in another document, the implementation of IHlink::Navigate turns around and asks the container for a moniker to the container itself using IHlinkSite::GetMoniker. The ActiveX hyperlink object then compares this moniker with the moniker it already contains, the one naming the hyperlink's target. If the two monikers are the same, the hyperlink knows that it refers to another location in the current document. If not, it must refer to a location in a different document.

If the ActiveX hyperlink object determines that it refers to a location within the current document, the container for that document must support IHlinkTarget. (It's the target for this hyperlink, after all.) The hyperlink gets a pointer to this interface by calling the container's IHlinkSite::QueryService method. If this hyperlink does not refer to a location in the container's current document, the hyperlink object calls IMoniker::BindToObject on the moniker it contains. For a hyperlink containing a file moniker with a filename such as REPORT.DOC, for instance, calling BindToObject will typically start Microsoft Word (because of the DOC extension) and hand it this file through IPersistFile. If the hyperlink contains a URL moniker such as http://www.acme.com/report.htm, it will fetch the HTML page identified by this URL and hand it to a web browser such as Internet Explorer. Whatever kind of moniker is involved, the initial interface the hyperlink requests on BindToObject is IHlinkTarget.

One way or another, the ActiveX hyperlink object now has a pointer to the IHlinkTarget interface of the target. The hyperlink object next invokes IHlinkTarget::Navigate, passing in the location string that this hyperlink stores. The source finds the correct information and causes it to be displayed. If this hyperlink is to another location in the current document, the current window displays the new information. If necessary, however, a new window is created and correctly positioned to present a smooth transition to the user, much as is done with OLE in-place activation. And although this brief description omits the details, the frame (if there is one) is kept informed about what's going on, and the browse context object is updated with the result of this navigation throughout the process of following the hyperlink.

This is similar to OLE linking using a composite moniker built from a file moniker and an item moniker. In that case, the file moniker identifies both the application and the document, while the item moniker passes the application a string that identifies a location within the document. By identifying a location within a document using a simple character string rather than an item moniker, the ActiveX Hyperlinks technology avoids the overhead of creating a moniker for the common case of hyperlinking to another location in the same document.

Integrating the browsing metaphor throughout their environment is likely to make users happy. Given what's just been described, however, it might leave software developers somewhat less pleased. Developers want a simple, powerful way to implement browsing, and although what we've seen so far is powerful, it's not especially simple. The implementor of a web browser or an application such as those in Microsoft Office might need a detailed understanding of the ActiveX hyperlinking architecture, but most programmers need only a straightforward way to add hyperlinks to their application. A simple hyperlinking API has been created to make this possible.

The primary purpose of this simple API is to make it easy to navigate to the target of a hyperlink. The API's small group of functions listed on the following page are focused around this goal.

- The **HlinkSimpleNavigateToString** function causes a jump to another location, presenting the user with a new set of information. The caller passes in a string, such as a filename or a URL, along with a location string and a few more parameters. The implementation of this call creates a moniker from the string (using MkParseDisplayNameEx, described in "A Generalized Approach to Naming," page 151) and creates an ActiveX hyperlink object containing that moniker and the location. It then navigates to the object this hyperlink identifies. A simpler version of this call, **HlinkNavigateString**, performs the same task but provides defaults for most of the parameters.
- The **HlinkSimpleNavigateToMoniker** function, like HlinkSimpleNavigateToString, causes a jump to another location. Its parameters are the same, too, except that the caller passes in a moniker instead of a string. A simpler version, called **HlinkNavigateMoniker**, provides defaults for most parameters.
- The **HlinkGoBack** function causes a jump to the previous location in the navigation stack maintained by the browse context object. This call works only if it is made by an application hosted in a hyperlink-aware frame, such as Internet Explorer. (This limitation exists because the implementation of this call relies on the frame to locate the browse context object — without it, there's no way to find the navigation stack and hence no way to go back.)
- The **HlinkGoForward** function causes a jump to the next location in the navigation stack. Like HlinkGoBack, it works only when made by an application hosted in a hyperlink-aware container.

Using these calls, any application can follow hyperlinks to any other application that supports the basic interfaces required to be a hyperlink target. An ActiveX control, for example, might present the user with a button that represents a link to a predefined Word document. When the user clicks on this button, the control can call HlinkNavigateString with the name of the file, and a hyperlink jump to that document will immediately occur. Rather than understanding and implementing calls to the underlying objects and interfaces, a developer can achieve the most commonly used features of ActiveX hyperlinking with a minimum of effort.

We work in a great business. Where else could new technology as transforming as that of the Internet and the World Wide Web so quickly become an important part of our lives? The downside of this enormous rate of change, of course, is that we're constantly forced to learn how to live with and use these new technologies. Sometimes this is easy. For the average software professional, learning to use a web browser takes less than five minutes. Sometimes, though, it's not so easy. Understanding the ActiveX technologies that underlie Microsoft's approach to the Web, for example, requires a firm grasp of COM, persistence, monikers, OLE, ActiveX controls, and more. It also requires understanding basic web technologies such as URLs and HTML. The reward for all this effort should be substantial, however. Whatever can be said about the tremendous amount of Internet hype — and it has frequently exceeded the bounds of rationality — one thing is sure: the Internet and the Web will be part of our lives for quite some time.

So, too, will ActiveX and OLE. COM and the technologies it has spawned have worked their way into the very fabric of Windows and Windows NT, two systems whose popularity is not declining. Understanding the ramifications of COM is essential to understanding software in the Microsoft world. And, one way or another, understanding the Microsoft world is important for nearly everyone in this exciting business we're in.

Although COM has since been applied to many other problems, it was originally created as part of a mechanism for creating compound documents. In some ways, the ultimate compound document is the World Wide Web. It shouldn't be surprising, then, that COM has been applied to the problem of web access, too.

This section includes the following topics:

Think for a moment about what happens when a web browser downloads a typical HTML page from a web server. When the information is received, the browser interprets the HTML and displays the page to the user. In older browsers such as Microsoft's Internet Explorer 2.0, the code for displaying HTML pages was built into the browser itself. As browsers came to be used to display more than just HTML, however, they needed a general way to load code on demand to handle any kind of information. If the user downloads a file in Adobe Acrobat format, for instance, the browser must be able to load the correct code to interpret that file and display the information. ActiveX Documents defines this sort of relationship—one application acting as a frame for another. It makes sense, then, to build a web browser using this technology, which is exactly what's done in Internet Explorer (IE) 3.0.

IE 3.0 separates generic browser functionality — navigating to a link, going forward and back, and so on — from the intelligence required to load, display, and manipulate particular kinds of information. The user sees one cohesive application, but the browser is actually built from several pieces, as shown in Figure 11-4. (Some of the relationships among the components are slightly simplified in the figure.) The smallest piece is the Internet Explorer *frame*, implemented in IEXPLORE.EXE. This simple piece of code does little more than provide a host process for the Internet Explorer *Web Browser object* (once known as the *shell document viewer*), implemented in SHDOCVW.DLL. This object provides generic browser functionality, and it communicates with the frame through various COM interfaces (the details of which aren't included here). The Web Browser object has no knowledge at all of HTML documents or any other sort of displayable information. What it does know how to do, however, is to act as an ActiveX Documents container. By loading the appropriate ActiveX Documents server, the Web Browser object can let the user see and work with many different types of information.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-4. Microsoft's Internet Explorer 3.0 is built from separate components glued together using COM.

Its deconstructionist look notwithstanding, IE 3.0 is still a web browser, and a key part of its function is displaying HTML pages. When asked to display an HTML page, the Web Browser object loads the *HTML viewer*, shown in Figure 11-4. This viewer, implemented in MSHTML.DLL, is an ActiveX Documents server that contains all the code required to display and work with HTML documents. Figure 11-5 shows an HTML page displayed using IE 3.0's frame, Web Browser object, and HTML viewer. All these components work together to present the user with the familiar, seamless look of a web browser.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-5. An ordinary HTML page displayed using Internet Explorer 3.0.

Because the Web Browser object is an ActiveX Documents container, it can also load and display anything that knows how to act as an ActiveX Documents server. Why not load an Excel file, for instance, into a web browser? Excel is capable of acting as an ActiveX Documents server, as shown earlier in the Binder example. Accordingly, IE 3.0's Web Browser object can load Excel and a spreadsheet the same way it loads the HTML viewer and an HTML document. Figure 11-6, an Excel spreadsheet displayed using IE 3.0, illustrates how this looks to a user. Because the ActiveX Documents technology exposes the full functionality of an embedded application, hosting Excel within IE 3.0's Web Browser object in no way limits what the user can do. This spreadsheet can be directly edited just as if the user were working with a stand-alone instance of Excel.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11-6. An Excel spreadsheet displayed using Internet Explorer 3.0.

To the Web Browser object, both the HTML viewer and Excel look identical: they're ActiveX Documents servers. This ecumenical approach to browsing means that a web server can store information in various formats (not only HTML pages) and then let the browser load the appropriate code to work with that information. For example, if an Excel spreadsheet is stored on a web server on the Internet, an ActiveX Documents–enabled browser can let its user click on a reference to that page and then automatically load Excel (assuming that Excel or a simpler ActiveX Documents–enabled Excel viewer is available on the browser's machine) and display the spreadsheet as an ActiveX document within the browser. A user can now use one approach — browsing — to access HTML pages on the Web, application-specific files on a local hard drive, and nearly anything else.

The Web Browser object also qualifies as an ActiveX control, which means that it can be plugged into any control container.

When you start Microsoft Windows, the user interface you see is provided by an application called the *shell*, which provides you with a way to access other applications and files on your machine. In Windows 95 and Windows NT 4, the standard shell presents a desktop metaphor, allowing you to work with the contents of your machine through folders and files in those folders. A web browser presents a different metaphor. Here you navigate through data and applications by following *hyperlinks* between documents, moving forward and back as needed. Given the popularity of browsing, integrating this new metaphor into the user interface is very desirable.

Given the structure of Internet Explorer, it's also very simple to accomplish. In IE 3.0, a generic browser (the Web Browser object) is loaded into a simple frame. Because that browser is an ActiveX Documents container, it allows users to access all sorts of information using the browsing metaphor. To let users access their systems as a whole using the browsing metaphor, then, all that's required is to modify the Windows shell so that it functions more like IE 3.0 and can then serve as a frame for the Web Browser object. In practice, this means adding support for a few more COM interfaces to the shell, not an especially onerous task. The shell itself can then host the Web Browser object in a natural way, and users can access information using this tool's generic navigation facilities. Files and applications on the local disk, a local network, or the Internet can all be browsed directly from the shell — there's no need for a special web browser application. And through the generic interfaces of ActiveX Documents, other applications can be loaded into that frame to work with other kinds of data, not just HTML pages.

This is exactly what happens in Internet Explorer 4.0. By supplying a new Windows shell, one that is capable of acting as a frame for the Web Browser object, the browsing metaphor can be applied throughout the user's environment. This is more than just a benefit for users — it's also a great example of the power of components. Code originally built for one application, a web browser, can be reused in a very general way.

Truly integrating browsing throughout the Windows user interface requires more than this, however. Browsing depends on the ability to create links among documents and to follow those links from one document to another. A traditional browser allows hyperlinks from one HTML document to another, but applying browsing more generally implies the ability to create more general links as well. A user might want to create a link from a PowerPoint presentation to a Word document, for example, or from a Word document to an Excel spreadsheet. Ordinary HTML hyperlinks aren't enough. To address this problem, the ActiveX family includes a technology called ActiveX Hyperlinks, which allows the creation of hyperlinks between all sorts of documents, not just HTML documents. The ActiveX Hyperlinks technology is already supported by IE 3.0's Web Browser object. (For details, see "ActiveX Hyperlinks," page 305.)

Once the Windows shell itself lets you browse the Web, the need for a separate web browser application becomes less apparent. But while web browsers as such might one day fade into the mists of history, that day hasn't yet arrived. And even if browsers per se vanish, components such as the Web Browser object and the HTML viewer will survive. Like spreadsheets, word processors, and other applications, these components provide functions that are useful to other programs as well as to people. All that's required is for these components to expose a set of COM objects with appropriate interfaces that clients can use to access the components' services. In Internet Explorer 3.0, all of these interfaces are defined as dual interfaces, allowing easy access by clients written in Microsoft Visual Basic and similar languages as well as by C++ clients.

Internet Explorer 3.0 has two components that provide programmability: the Web Browser object, providing generic browsing capabilities; and the HTML viewer, with its HTML-specific functionality. The Web Browser object is typically driven from the outside by, say, a Visual Basic program that uses this object to locate a particular document. To make this possible, the Web Browser object exposes methods that correspond to a user's actions, such as the following:

® The **Navigate** method is used to move to a new location specified by a hyperlink.

® The **GoBack** method is used to move to the previous location in the history list.

® The **GoForward** method is used to move to the next location in the history list.

® The **Refresh** method refreshes the current view by reloading the document.

Like most objects accessed through dual or dispatch interfaces, the Web Browser object also has properties. This object's properties include the following:

® The **Type** property returns the type of the currently loaded ActiveX Documents server, such as HTML or Excel.

® The **Busy** property indicates whether an activity such as a document load is in progress.

® The **Document** property returns a pointer to the IDispatch interface of the ActiveX Documents server for the currently loaded document. If an HTML document is loaded, for example, this property returns a pointer to the IDispatch interface of the HTML viewer. If an Excel spreadsheet is loaded, it returns a pointer to Excel's IDispatch interface. Using this pointer, a client of the Web Browser object can access the methods made available by the currently loaded ActiveX Documents server, whatever it happens to be.

The Web Browser object can also send events, such as OnDownLoadComplete, an event indicating that the current page has been completely received. As with all events, the creator of a program driving the Web Browser object can write a subroutine that is called when this event is received.

Unlike the Web Browser object, the HTML viewer is typically driven from "inside." The viewer might, for example, load an HTML document containing an embedded script. This script then executes, making requests of objects within the HTML viewer as needed. The viewer supports several objects, arranged in a hierarchy. A script can directly access the topmost object in this hierarchy, the Window object, and then acquire access to objects below it through the Window object's properties.

The Window object represents the browser window that the user sees. Its methods include these three:

® The **Alert** method displays a simple message box.

® The **Prompt** method displays a message and prompts the user for a reply.

® The **Navigate** method causes a jump to a new location identified by a URL.

The Window object also has several properties, some of which return references to objects lower in the hierarchy. These properties include the following:

® The **History** property returns a reference to a History object containing a list of visited locations.

® The **Frames** property returns an array of the window's current frames.

® The **Document** property returns a reference to the current Document object.

Finally, the Window object is able to send two events: onLoad, sent when a page is loaded; and onUnload, sent (not surprisingly) when a page is unloaded.

After the Window object, the Document object is probably the most important for creators of scripts. This object, located using the Window's Document property, represents the currently loaded HTML document. Its methods

include Write, which writes text such as HTML code, and Open and Close, for opening and closing new documents. Among the Document object's many properties are bgColor, which sets a page's background color; linkColor, which sets the color for links on the page; and vlinkColor, which sets the color for links that the user has visited.

The Window object, the Document object, and all the other objects implemented by the HTML viewer can be accessed by scripts embedded in HTML documents that the viewer loads. If there were only one possible choice for a script language, it might make sense to build support for it into the HTML viewer itself. Several options for scripting languages are available, however, which suggests that a more general solution would be useful. That general solution, called ActiveX Scripting, is what the HTML viewer uses to execute scripts, and it's described next.

The HTML viewer's object model is patterned after the model exposed by Netscape Navigator. This makes it straightforward to create scripts that work with both Navigator and Internet Explorer.

Microsoft Office 97/Visual Basic Programmer's Guide
Published by Microsoft Corporation

This chapter shows you how to use Microsoft Office 97 to develop applications that retrieve, publish, and share information on the Internet or a local area network (LAN). For example, you can create applications that display Hypertext Markup Language (HTML) documents, or you can publish or share information located on a Web server. You can also create hyperlinks that you click to open Microsoft Office documents or objects located on a local hard disk or a LAN.

Chapter 15: Developing Applications for the Internet and World Wide Web

This chapter includes the following topics:

Residential Funding Corporation (RFC), a wholly-owned, indirect subsidiary of General Motors Acceptance Corporation (GMAC), is a respected leader in the mortgage finance industry. Supported by the financial strength and worldwide name recognition of General Motors, RFC brings expertise in securitization, balance sheet management, funding, technology and direct capital markets distribution to the table.

To help its customers compete effectively in the marketplace, RFC offers a wide range of innovative products and services. These offerings enable RFC customers to align themselves as market leaders with broad diversification opportunities. RFC products and services are fully customizable and are accompanied by RFC's unparalleled customer support.

Given their commitment to customer support, RFC faced a challenge: controlling the costs of services while staying at the head of their industry in terms of service quality. In particular, customers call each day to ask about the status of their loans. Many RFC employees spend significant parts of their workday manually looking up status information. The status data is managed by an IMS application on a mainframe, and is extracted nightly into an Oracle database. RFC realized that they could offer even better customer service – could save significant amounts of time and money — if they could automate access to status information kept in the Oracle database. In particular, RFC wanted to enable Intranet access via Web browsers, and to key customers via a Voice Response Unit (VRUs).

{ewc MVIMG, MVIMAGE,!1315_01.bmp}

RFC chose to develop their loan status inquiry system using a 3-tiered client/server architecture, ActiveX components and Microsoft Transaction Server. RFC knew that their application would have to be secure, auditable, and support both Web and VRU interfaces. Developing their application as a set of ActiveX components simplified development — and ensured RFC that it would be easy to use the same status inquiry components in both the Web and voice response environments.

RFC chose to use MTS because it simplifies building server-based applications with ActiveX components, ensures fast response times, and guarantees transactional data integrity. RFC's components use ODBC drivers to access status data in the Oracle database, and store audit data in a Microsoft SQL Server 6.5 database. The SQL Server database also stores a record of any requests the user makes for additional information. All application logic remains safely behind RFC's firewall. More important, using MTS with ActiveX components was easy — RFC built simple 'single user' ActiveX components using the Visual Basic programming system 5.0 and added a few lines of code required by MTS for transaction control. To make the application available to both the VRU and web browsers, RFC simply 'dragged and dropped' the compiled components into the MTS Explorer.

Microsoft's Internet Information Server (IIS) Version 3.0 hosts the Intranet site. The status inquiry user interface is managed via IIS's Active Server Page (ASP) technology. ASPs make it easy for RFC's Webmasters to design an interactive Web pages for accepting user input and then submit the user's data directly to ActiveX components running under Microsoft Transaction Server (MTS) for computation. No complex CGI scripting is required. In addition, IIS automatically protects the privacy of all data exchanged with the user by using the Secure Sockets Layer (SSL) protocol. RFC also plans to make the loan status application available on the Internet at some point in the near future.

On the voice response side, RFC uses a product from Dialogic Corporation. Voice interaction is 'scripted' using applications written in the Visual C++ development system. The applications access RFC's ActiveX components via the native support for ActiveX in Visual C++. Adding the VRU support to their application required only a few extra days of programming.

Perhaps most important, RFC — working with Millenium Software, a solution provider — was able to build their application with two developers in about three weeks. "I was extremely pleased with how quickly a small team was able to solve a significant business problem by building an n-tiered application, reusing application components from multiple user interfaces. This application architecture, enabled by MTS, IIS and ActiveX components will be the model for future development at RFC. I am gratified that this project has demonstrated that these technologies can move RFC toward our business goals of reuse, rapid development, scalability and supportability," said Rick Greenwood, RFC's Chief Technology Officer. "From a system designer and developer perspective, we are finding that MTS and ActiveX components are truly easy to learn and program, especially compared to more traditional transaction and n-tiered environments," Mr. Greenwood added.

In some cases, all that's required to download code is to copy a single executable file from a web server to the browser's machine. In other cases, it might be necessary to copy more than one executable file along with one or more supporting files. To deal with this variability, the *Internet Component Download* service used by Internet Explorer defines three packaging schemes for downloaded code:

- A portable executable (PE), containing a single executable file with an extension such as OCX, DLL, or EXE.
- A cabinet file, identified by the file extension CAB. A cabinet file can contain one or more executables, all compressed into a single package and downloaded as a unit. It also includes an INF file that directs the installation process of the cabinet's files.
- A stand-alone INF file, containing only references to other files that should be downloaded. An INF file can contain URLs referring to files on a single machine or on several machines. It can also specify options for which files to download depending on the type of client platform making the request. For example, a request to download an INF file made from a Windows 95 system and the same request made from a Macintosh system might result in copying different binaries.

The filename specified in the CODEBASE attribute can optionally be followed by a version number. If it is, the file is downloaded only if this version number is more recent than any version of this file currently resident on the system.

When a browser such as Internet Explorer attempts to download the code for an ActiveX control, its real goal is to create one or more COM objects using that code. Ultimately, then, the browser must acquire a pointer to the IClassFactory interface of the control's class factory and call CreateInstance. The Internet Component Download service makes this very easy. When Internet Explorer encounters a CODEBASE attribute inside an *OBJECT* tag and decides to download the associated code, it needs to call only the single function CoGetClassObjectFromURL. Like CoGetClassObject (discussed in "Using a class factory," page 61), this function returns a pointer to a class factory. As its name suggests, the caller passes in a URL specifying where to find the code. This URL can name a portable executable, a cabinet file, or an INF file, and the browser takes this value directly from the CODEBASE attribute in the *OBJECT* tag. The caller can also pass in the CLSID from the tag's CLSID attribute or the MIME type of the object indicated by the CODETYPE attribute. (The MIME type is mapped to a CLSID using the system registry.) Making this single call causes the control's code to be copied to the browser's system (if it's not already present), verified as safe using WinVerifyTrust (discussed in the next section), and registered with the system registry. Once everything has been installed locally, CoGetClassObjectFromURL calls CoGetClassObject to return a pointer to the class factory of the new object.

Of course, the actual process is a bit more complex. The implementation of CoGetClassObjectFromURL relies on a URL moniker to accomplish the downloading, which means that a client making this call must implement IBindStatusCallback to receive progress notifications. The caller of CoGetClassObjectFromURL must also implement the ICodeInstall interface. This simple interface lets the client learn about any problems that crop up during the download and handle any necessary user interface issues. Also, once a component is downloaded, no automatic mechanism deletes it — it remains on that system's disk indefinitely. For the most part, however, clients such as Internet Explorer are shielded from the messy details of downloading objects.

Being able to download components as needed is a useful capability. By default, a downloaded Java applet is wrapped in a secure cocoon during execution. Because each applet has its own safe “sandbox” to play in, providing security in this way is sometimes called *sandboxing*. Unlike Java applets, however, ActiveX controls are binaries executing directly on the machine’s hardware. Although ActiveX controls have capabilities that sandboxed Java applets do not, they also offer more opportunities for mischief. A malicious developer could easily create controls that, say, reformat the hard drive of any machine that installs them. If users can’t have faith that a given control won’t damage their system, they can’t take the risk of downloading and running that control.

Creating that faith is the goal of the *Windows Trust Verification Services*. Through the single function call WinVerifyTrust, a user of this service can access one or more *trust providers*. In general, a trust provider can answer questions about whether a component can be trusted according to certain criteria. The initial release of this service includes only one choice, the Windows Software Publishing Trust Provider. This trust provider is able to answer the question that most concerns the potential user of a downloaded ActiveX control: was this control produced by someone I trust?

At first glance, this might appear to be the wrong question. What users really want to know is whether this control will damage their system, not who created it. Unfortunately, there’s no general way to determine this. The best users can do is assure themselves that the software was created by a trusted source and that it hasn’t been modified since its creation. This is similar to the faith users express when buying packaged software. If the box carries the name of Lotus or Microsoft or another reputable vendor, and if the shrink wrap on the package isn’t broken, users can feel confident that the software inside won’t intentionally damage their system. Providing this same kind of confidence is the goal of the Windows Software Publishing Trust Provider.

When Internet Explorer 3.0 downloads a component, that component might carry with it a *digital signature*. A digital signature is a byte string that can be used to verify that the associated information was actually provided by a specific entity. More than that, a digital signature also verifies that the information (in this case, the downloaded code) hasn’t been modified since the signature was affixed. In essence, the signature plays the role of both the company name on a software package and the package’s shrink wrap.

To allow others to verify its digital signature, a component carries with it another byte string called a *certificate*. When Internet Explorer calls WinVerifyTrust, it passes in references to both the newly downloaded control’s digital signature and its certificate. The trust provider examines both and returns an indication of success or failure. If the check fails or if the component is from an untrusted source, IE 3.0 informs the user and offers a choice of whether to proceed. Note that because a digital signature verifies that the associated information hasn’t been modified from its original form, it’s impossible to silently insert viruses into the code. By having the developer add a signature to a component and having the browser check that signature after downloading, a system is created whereby a user can have a high degree of faith in the component’s trustworthiness.

The details of how digital signatures work are beyond the scope of this book. For those who are familiar with the technology, the Windows Software Publishing Trust Provider uses PKCS #7 and X.509 version 3 certificates. For those who aren't, well, you can trust me on this.

Over time, concepts and techniques that allow us to build large applications have been discovered. Modularity, or structuring an application as independent modules, allows you to build complex systems from simpler parts and reuse software. Object-oriented concepts and the Microsoft Component Object Model (COM) provide a technique that allows you to write modular applications.

Components can give both modularity and natural distribution. When an application is structured as components, the individual parts can reside together in a single computer, or they can interact by using remote-procedure calls across a network.

Structuring an application into independent components can create problems with managing the components. Monolithic programs fail and restart as a unit. But, with a modular system, the failure of one component must not corrupt the others. There must be a way to isolate faults and limit fault propagation. Transactions provide modular execution, simplify and automate fault handling, and provide a simple conceptual execution framework for both implementers and users.

The user thinks of a transaction as a single change event that either happens or doesn't happen. Implementers think of a transaction as a programming style that allows them to write modules that can participate in distributed computations. Consider a transfer of money from one bank account to another. The implementers and the users want to make sure that either both accounts change or neither changes. It is hard to make this work in a distributed system — computers can fail and messages can be lost. Transactions provide a way to bundle a set of operations into an atomic execution unit.

The atomic all-or-nothing property is common; it appears in many everyday proceedings. When several parties enter into a contract, an escrow officer coordinates the transaction and collects the signatures of each party to the contract. The contract is final only when the escrow officer announces that everyone has signed. A minister conducting a marriage ceremony first asks the bride and groom, "Do you take this person to be your spouse?" If they both respond "I do," the minister pronounces them married. A director on a movie set first asks, "Ready on the set?" If all respond yes, the director then calls, "Action!" A helmsman on a sailboat preparing to tack first asks the crew, "Ready about?" If they all respond yes, then the helmsman shouts, "Helm's a'lee!" and turns the boat.

These scenarios illustrate the basic principle of a transaction: several independent entities must agree. If any party disagrees, the deal is off. Once they agree, the transaction can occur. The Microsoft Distributed Transaction Coordinator (MS DTC) performs this transaction coordination role for the other components of the COM architecture.

In MS DTC terminology, the director is called the transaction manager. The participants in the transaction that implement transaction-protected resources, such as relational databases, are called resource managers.

An application begins a transaction by calling the transaction manager's **BeginTransaction** method. This creates a transaction object that represents the transaction. The application then calls the resource managers to do the work of the transaction.

The application's first call to each resource manager identifies the application's current transaction. For example, if the application is using a relational database, it calls the Open Database Connectivity (ODBC) interface, which associates the transaction object with the ODBC connection. Thereafter, all database calls made over that connection are performed on behalf of the transaction until the transaction is ended.

When a resource manager first does work on behalf of a transaction, it *enlists* in the transaction by calling the transaction manager. As the transaction progresses, the transaction manager keeps track of each of the resource managers enlisted in the transaction.

Typically, the application completes the transaction with a **Commit** transaction method. If the application is unable to complete, the application calls the **Abort** transaction method, which undoes the transaction's actions. If the application fails, MS DTC aborts the transaction.

When the application successfully completes the transaction's work, it calls the MS DTC to *commit* the transaction. MS DTC then goes through a *two-phase commit protocol* to get all of the enlisted resource managers to commit. The two-phase commit protocol ensures that all the resource managers commit the transaction or all abort it. In the first phase, the MS DTC asks each resource manager if it is *prepared* to commit. If all participants say yes, then in the second phase MS DTC broadcasts the commit message to all of them. If any part of the transaction fails, if a resource manager fails to respond to the prepare request, or if a resource manager responds no, then MS DTC notifies all of the resource managers that the transaction aborted.

Transaction managers are a key part of most database systems. Transaction managers are also an optional part

of some operating systems. Microsoft believes that transactions are essential for distributed applications — transactions provide modular execution, which complements COM's modular programming. So, Microsoft implemented transaction management software for both the Microsoft Windows 95 and Microsoft Windows NT operating systems.

Combining the transaction concept with COM required innovation. Traditional transaction systems required considerable skill to install and manage. The challenge of integrating MS DTC with the Microsoft operating systems was to automate installation, management, and use. Many of the concepts and techniques had to be reinvented for the new client/server, object-oriented, and visual management environments.

In its first release, MS DTC works with one resource manager: Microsoft SQL Server. It also operates with several transaction processing monitors, including Encina Top End, and Novell Tuxedo. MS DTC implements the OLE transaction interfaces. All OLE transaction interfaces are public so that any resource manager can become an OLE transaction resource manager. In the future, Microsoft and other software companies will add other transactional resource managers, such as distributed object systems, transactional file systems, transaction queuing systems, and workflow management systems.

Transactions provide the ACID (atomicity, consistency, isolation, durability) properties.

® Atomicity

A transaction either commits or aborts. If a transaction commits, all of its effects remain. If it aborts, all of its effects are undone. For example, in renaming an object, the new name is created and the old name is deleted (commit), or nothing changes (abort).

® Consistency

A transaction is a correct transformation of the system state. It preserves the state invariants. For example, by adding an element to a doubly linked list, all four forward and backward pointers are updated.

® Isolation

Concurrent transactions are isolated from the updates of other incomplete transactions. These updates do not constitute a consistent state. This property is often called *serializability*. For example, a second transaction, traversing the doubly linked list mentioned in the consistency example, will see the list before or after the insert, but it will see only complete changes.

® Durability

Once a transaction commits, its effects will persist even if there are system failures. For example, after the rename in the atomicity example, the object will have the new name even if the system fails and reboots right after the commit completes.

It is up to the application to define consistency and bracket its computation with **BeginTransaction** and **Commit** transaction methods to delimit these consistent transformations. Transactional resource managers provide consistent, isolated, and durable transformations of the objects they manage. MS DTC manages transactions that involve multiple resource managers, even those distributed among multiple computers. MS DTC creates transaction objects, tracks migration of transactions among resource managers, and implements the two-phase commit protocol to make these transactions atomic and durable.

We have already discussed the end user's view of transactions; transactions are ACID execution units that either commit or abort. If a transaction commits, its effects are durable. If a transaction aborts, its effects are undone. Application programmers, resource managers, and transaction managers cooperate to implement the ACID properties. Let's look at the role of each of these participants.

This section includes the following topics:

The discussion so far has assumed that all the applications and resource managers are on a single computer. MS DTC also supports transactions distributed across two or more Windows systems. Each system has a local transaction manager. All applications and resource managers talk to their local transaction managers. The transaction managers cooperate to manage transactions that span systems.

When a transaction first visits a new system (when the first request tagged with the transaction arrives at that system), the two systems involved in the request establish a relationship. The system making the request informs its local transaction manager that the transaction has an *outgoing* relationship to the transaction manager on the second system. Similarly, the system receiving the request informs its local transaction manager that the transaction has an *incoming* relationship with the transaction manager on the first system.

These outgoing-incoming relationships form a tree of transaction manager relationships called the transaction's *commit tree*. The enlisted resource managers are also members of this commit tree; they have an outgoing-incoming relationship to their local transaction manager.

When a distributed transaction commits or aborts, the prepare, commit and abort messages flow outward on the commit tree. Any node of the tree can unilaterally abort a transaction anytime before it agrees to the prepare request sent at phase one. Once a node has prepared, it remains prepared and in doubt until the commit coordinator tells it to commit or abort the transaction. The root transaction manager of the commit tree is the global *commit coordinator*. It makes the decision to commit or abort the transaction and is never in doubt.

If a computer fails and then restarts, the transaction manager at that computer will determine the outcome of all in-doubt transactions. The transaction manager reads its log file to determine the outcome of transactions for which it was the commit coordinator. For incoming transactions from other systems, the transaction manager reads the log file to determine whether it was previously notified of the transaction's outcome. For incoming transactions that remain in doubt, the transaction manager queries the incoming transaction manager to learn the transaction's outcome. The transaction manager also responds to queries from other transaction managers regarding in-doubt outgoing transactions sent to them. This is similar to the protocol that transaction managers and resource managers follow at restart. The transaction manager determines the outcome of each in-doubt transaction and, when asked, tells the resource managers the transaction's outcome.

In-doubt transactions are especially troublesome for distributed transactions. System or communication failures can leave transactions in doubt for a long time. While the transaction is in doubt, the resources modified by the transaction remain locked and unavailable to others. MS DTC provides a way for the system operator to resolve transactions that remain in doubt for too long. The operator can use a graphical management interface at the commit coordinator system to determine the transaction's outcome. The operator also can use the graphical management interface at the in-doubt system to force the in-doubt transaction to commit or abort. When the systems reconnect, MS DTC detects these operator actions and flags inconsistent actions.

Transactions are ACID (atomic, consistent, isolated, durable) modules of execution. They complement COM's program module structure. The Microsoft Distributed Transaction Coordinator (MS DTC) provides a transaction manager for each computer that manages transactions. Applications call the transaction manager to *begin* a transaction. **BeginTransaction** returns a transaction object. The application includes the transaction object with requests to resource managers. When a resource manager first begins working on a transaction, it enlists in the transaction. When the application has made a consistent transformation of the state, it asks the transaction manager to *commit* the transaction with the **Commit** transaction method. If the application cannot complete the transaction, the application program aborts it by using the **Abort** transaction method. If the application fails or a participating resource manager fails, then MS DTC aborts the transaction.

MS DTC uses a *two-phase commit* algorithm in which (1) the transaction manager requests each enlisted resource manager to *prepare* to commit, and (2) if all successfully prepare, the transaction manager broadcasts the commit decision. If any resource manager cannot prepare, the transaction manager broadcasts an abort decision to everyone involved in the transaction. While a resource manager is prepared, it is *in doubt* about whether the transaction committed or aborted. The transaction manager keeps a sequential *log* so that its commit or abort decisions will be durable. If a resource manager or transaction manager fails, it reconciles in-doubt transactions when it reconnects.

For distributed transactions, each computer has a local transaction manager. When a transaction works at multiple computers, the transaction managers track *incoming* and *outgoing* transactions. Each transaction manager performs all the enlistment, prepare, commit, and abort calls for local resource managers (on that computer). When committing a transaction distributed among several computers, the transaction manager sends prepare, commit, and abort messages to all its outgoing transaction managers. When a transaction manager is in doubt about a distributed transaction, the transaction manager queries the incoming transaction manager. The root transaction manager is never in doubt. If an in-doubt transaction persists for too long, the system operator can force the transaction to commit or abort.

The application programmer's model of transactions is quite simple: applications either succeed or fail. The application begins a transaction by getting a transaction object. All subsequent work is associated with that transaction object. When the program reaches a consistent state, it calls the **Commit** transaction method. If the commit succeeds, the transaction is durably committed. If the commit fails, the transaction is aborted. If the program finds that it cannot complete the transaction, it may call the **Abort** transaction method to undo the transaction's effects. This is a simple way to clean up complex failure cases.

If the application fails before it commits the transaction, the transaction manager will abort the transaction and tell each enlisted resource manager to undo the transaction's effects. If a computer or resource manager fails, the transaction will also be aborted. Once the transaction has successfully committed, the resource managers and the transaction manager will ensure that the transaction's effects are durable, even if there are subsequent failures.

When a resource manager first comes on the scene, it contacts its local transaction manager to declare the resource manager's presence. Then the resource manager waits for execution requests from applications. When a request arrives tagged with a new transaction object, the resource manager enlists in the transaction. By enlisting, the resource manager ensures that it will get callbacks from the transaction manager when the transaction commits or aborts. The resource manager then performs the transaction's requests. For example, the transaction might insert, delete, or update records in a relational database. The resource manager keeps enough information to allow it to either undo or redo the transaction's work on those resources. There are many ways to do this; keeping versions of data or keeping a log (journal) of the changes are two common techniques.

When the application commits the transaction, the transaction manager initiates the two-phase commit protocol. The transaction manager first asks each enlisted resource manager if it is prepared to commit the transaction. The resource manager must *prepare to commit* — it readies itself to either commit or abort the transaction.

Typically, the resource manager records the old and new data in stable storage so that the resource manager can recover it even if the system fails. If the resource manager cannot prepare successfully, it informs the transaction manager that it cannot prepare and the transaction manager aborts the transaction. If the resource manager can prepare, it tells the transaction manager and waits for the transaction manager's decision whether to commit or abort the transaction.

Once prepared, a resource manager must wait until it gets a commit or abort callback from the transaction manager. Most transactions commit, although a few transactions abort. Typically, the entire prepare and commit protocol completes in a fraction of a second. If there are system or communication failures, the commit or abort notification may not arrive for minutes or hours. During this period, the resource manager is *in doubt* about the outcome of the transaction. It does not know whether the transaction committed or aborted. While the resource manager is in doubt about the transaction, it keeps the data modified by keeping the transaction locked, thereby isolating these changes from any other transactions.

The discussion so far shows how distributed transactions are made atomic in a fault-free environment. Now we consider how MS DTC helps resource managers make transactions atomic and durable when a resource manager fails. If a resource manager fails and then restarts, the resource manager must reconstruct the committed state of the resources it manages. The reconstructed state must reflect all of the effects of committed transactions and none of the effects of aborted transactions. When a resource manager fails, all of its enlisted transactions are aborted except for those that prepared or committed prior to the failure. When the resource manager restarts, it asks the transaction manager about the outcome of the in-doubt transactions in which it enlisted. The transaction manager tells the resource manager the outcome of each in-doubt transaction, and the resource manager commits or aborts these transactions accordingly.

In summary, the two-phase commit protocol and the resource managers combine to make transactions atomic and durable.

The transaction manager is the manager of transaction objects. It creates transaction objects and manages their atomicity and durability. Applications ask the transaction manager to create a transaction object by calling the transaction manager's **BeginTransaction** method.

When a resource manager first participates in a transaction, it calls the transaction manager to enlist in the transaction. The transaction manager tracks the resource managers that enlist in the transaction. Later, the application commits or aborts the transactions, or the transaction is aborted by a resource manager or a failure.

Commit and **Abort** are additional transaction methods on transactional objects. When asked to commit a transaction, the transaction manager initiates a two-phase commit protocol. During phase one, it asks all enlisted resource managers to prepare. During phase two, the transaction manager tells the resource managers whether the transaction committed or aborted. The two-phase commit protocol has many optimizations, including the read-only optimization and the transfer-of-commit optimization. MS DTC implements some of these optimizations, but the functionality remains the same: atomicity and durability.

The transaction manager keeps a *log* in safe storage on disk. The log is a sequential file that records transaction events. The transaction manager records transaction starts, enlistments, and commit decisions in the log. During normal processing, the transaction manager only writes the log. If the transaction manager fails, it reconstructs the transaction's most recent state by reading the log. So, the transaction manager uses the log to make its state durable.

The transaction manager also provides an operations interface to manage transactions. It maintains performance counters that can be displayed by using the system performance monitor. It records important operational events in the system log. These events can be displayed by using the system event viewer. It has a graphical management interface that is integrated with SQL Enterprise Manager. The graphical management interface lets the operator configure the system, view transactions, and abort or commit in-doubt transactions.

Remote Data Objects (RDO) is an incredibly useful and powerful tool for Microsoft Visual Basic client/server developers, but using it fully has proven to be a challenge for many of these same developers. While it is possible to gain speed improvements by simply porting existing code and libraries to this new interface, this approach fails to tap the real power and performance that can be gained by putting some of the more complex techniques and features to work in your code. As is usually the case, there is a relationship between creating complex code and the performance benefits that are actually gained by doing so.

Now you can use the power of Data Access Objects (DAO) version 3.1 to leverage existing code and the power of a tight, fast interface to Microsoft SQL Server. With DAO 3.1, the existing object model has been extended to include the features and enhancements that are present in RDO 1.0 today. What this means to developers is that by using DAO 3.1, the same code syntax they use today will, with only slight modifications, bypass the familiar Microsoft Jet functionality we all know and love and go directly to Open Database Connectivity (ODBC) for high-speed access to Microsoft SQL Server.

What this really means is that many developers who currently program against Microsoft Jet databases will be using the same techniques to access data on Microsoft SQL Server. This article will show how to maximize the benefits of programming for Microsoft SQL Server, whether you are using RDO 1.0 today or upgrading to DAO 3.1 later.

To begin to leverage the power of ODBC and Microsoft SQL Server, you need to fully understand the cursors and their role in efficient scalable data access.

Cursors aren't really as enigmatic as we sometimes think. In fact, most people have a basic idea of what they are. Of course, most database developers use them, albeit unwittingly, just about every time we use DAO, RDO, or the ODBC API function. These interfaces all use cursors in some form. Whenever our applications require data access and we request a **Dynaset**, **Recordset**, or **Resultset** to be opened, we are really receiving a type of cursor from the interface. These interfaces can have their own cursor library or they can use the cursors provided by the data source we are accessing. So as a user of a cursor, you aren't creating the cursor directly. You are requesting it from a service provider such as a relational database management system (RDBMS) or a cursor library such as that included in the ODBC API. Now that we know where we get cursors from, let's talk about what cursors can do.

Because all of us probably have different ideas of what cursors are, I want to set down a simple working definition for clarity in this discussion. A cursor is defined here as being the manipulator of a set of data. This data is prepared by a service, exists within the address space, and uses the resources of the owner of the cursor. A cursor manages this data and has the ability to retrieve a portion of that data for a user of the cursor. A request by the cursor's user to retrieve a piece of data is called *scrolling*.

Because there can be many ways for a cursor to interact with the service (the data provider) and the user (who makes requests about the data), there are many types of cursors that can be created. Fortunately, there are only a few major distinctions in cursor types. These distinctions can overlap a great deal, so the variety of cursors is more like a collection than a specific list. So, rather than trying to list the distinct varieties, I will start by outlining the distinguishing factors. As you read, you will begin to see how the distinctions merge.

Updatable and Nonupdatable Cursors

Updatable cursors give the user the capability to make changes to the data in a cursor and have those changes propagated back to the data provider by the cursor. You can think of this as having write privileges on the original data. In a read-only (unupdatable) cursor, any changes to the data in a cursor cannot be propagated back to the data provider. These kinds of cursors offer generally better performance because they allow the data provider to offload the data to the cursor once and then essentially forget about it and continue servicing other requests, free from concerns about concurrency problems. Both updatable and nonupdatable cursors support scrolling.

Scrollable Cursors

Cursors in all forms use the concept of a current record. A *current record* is the unit of information that is currently available for transfer from the cursor to the user of the cursor. Cursors that allow the user of the cursor to request that a piece of data be made the current record more than once are called *scrollable cursors*. Cursors that don't have this ability are called *nonscrollable cursors*; they can only provide data in a first-in, first-out (FIFO) format as data is requested. Consequently, a nonscrollable cursor is an efficient kind of cursor because it requires only the resources used to hold the data until the user of the cursor asks for the data. After the data that makes up the current record is transferred to the user, the cursor can release the resources that were used to hold that piece of data. In addition, a nonscrollable cursor doesn't require any logic about the data it holds. It only holds the data and gives it up when asked. In most implementations, when a nonscrollable cursor is requested by the user, the cursor provider (the service from which a user requests a cursor) doesn't use a cursor and instead connects the data requester to the data provider somewhat directly. With this in mind, we will limit our discussion to those types of cursors that do support scrolling.

For a cursor to be scrollable, it must have logic that will allow the user to request by position the data presented in the current record. When the cursor receives a request to change the data in the current record, it must retrieve the data from its private store and populate the current record for the user. Typical requests that are supported include the requests to move to the beginning or end of a cursor as well as to move one unit of information forward or backward within the cursor. You may recognize these as the **MoveFirst**, **MoveNext**, **MovePrevious**, and **MoveLast** methods present in **Table**, **Dynaset**, **Recordset**, or **Resultset** objects. In effect, these methods are how a user initiates scroll requests from a cursor.

When one compares scrollable and nonscrollable cursors based on performance, the nonscrollable cursors usually win by a wide margin. It becomes easy to see why when you consider the logic involved with maintaining data versus just passing the data around. In an ideal world, the user of a cursor would want to use these different types of cursors at different times within the same application. Unfortunately, presenting an interface that works with both scrollable and nonscrollable cursors can be challenging. For quite a while, this issue had been neglected. Interfaces for data access provided either scrollable cursors (such as DAO) or nonscrollable cursors (such as DB-Library). But now, you can actually choose at run time whether to use a scrollable or

nonscrollable cursor. Using RDO, you have an optional constant that can be used when calling the **OpenResultset** method. This constant is defined in the RDO interface as **rdOpenForwardOnly**. By including this in the options used when calling the **OpenResultset** method, a nonscrollable cursor is created. With DAO 3.1, there is an additional type of recordset that can be opened, **dbOpenForwardOnly**. What this means programmatically is that on a resultset or recordset opened with this option, the only navigation method allowed is the **MoveNext** method. This allows the application to retrieve the data as fast as it can be transferred from the data provider. For situations where no updates are expected, and the data is accessed in a simple loop, this is the fastest way to get the data.

Cursor Keysets

Scrollable cursors can retrieve and manage the data elements that make up the individual records from the data provider in several possible ways. One way is to use a keyset. You establish a keyset when the cursor retrieves only the unique keys for the records from the data provider. These record identifiers are called *keys*. When a specific record is requested, the cursor uses the key for that record to get the full contents of the record. The collection of these keys is called a *keyset*. Of course, there are many ways to define and use a keyset in a cursor. We will talk about some of these variations a little later.

An important aspect of using a keyset-driven cursor is that it can be faster than a standard scrollable cursor because it has less data to manage at any one time. A standard scrollable cursor would have to store the entire contents of each record in the set of records. By using a keyset, the cursor only has to keep track of the keys for each record. The key for a record is almost always smaller than the other elements of the record combined.

Cursor Membership

Whether or not a cursor uses keysets, all cursors must have rules regarding membership. *Membership* in a cursor is defined by the point in time when the set of records in a cursor becomes fixed. In a cursor that doesn't use keysets, the membership is usually fixed at cursor load time. That's because the operation involved with retrieving and storing each record is so expensive. In a keyset-driven cursor, this cost is lessened, so it is common for keyset-driven cursors to require membership only at scroll time. Of course, if the cursor is *dynamic*, then the rules change.

Dynamic Cursors

A dynamic cursor manages only a portion of the entire recordset at any one time, and it retrieves only the first *n* records at load time. When the cursor owner requests a record outside the scope of the currently loaded recordset, the cursor loads an additional *n* records. There is some variation in exactly when the loading of records is done, but this approach is generic enough for discussion.

So cursors can be dynamic and use keysets or dynamically load static data. When a cursor dynamically loads the keyset, it is called a *mixed cursor*. Mixed cursors are generally a good common ground when the application needs scrollable access to a large set of records.

When using RDO and Microsoft SQL Server, you have the advantage of using another type of cursor to further increase performance. Using RDO or DAO 3.1, you automatically have access to the ODBC cursor library. But with Microsoft SQL Server, you get access to the Microsoft SQL Server built-in cursor functionality. To take advantage of this, you need to set the **rdoDefaultCursorDriver** or **CursorDriver** properties. By using Microsoft SQL Server cursors, you can increase performance in several areas, mainly because the server is doing the caching required of a cursor, instead of downloading records to be cached at the workstation. With Microsoft SQL Server version 6.0 and later, you have full access to all the previously mentioned types of cursors, so you can choose any cursor options you like.

The main cursor types are exposed through RDO using the following flags on the **OpenResultset** method:

® **rdOpenForwardOnly**: opens a *forward-only* scrollable cursor (default).

® **rdOpenStatic**: opens a scrollable cursor with *static* data.

® **rdOpenKeyset**: opens a scrollable cursor using a *keyset*.

® **rdOpenDynamic**: opens a scrollable cursor using a *dynamic keyset* (also known as a *mixed cursor*).

Asynchronous operation has always been a challenge for Visual Basic developers. Using the new features of RDO and DAO 3.1, the database perspective on this changes radically. The following section outlines a common way to accomplish asynchronous query execution in Visual Basic using RDO syntax. The DAO 3.1 syntax would be very similar and mostly redundant.

A common way to create the disjoint required for asynchronous execution is to use a timer control. The timer can be set to fire at specific intervals. To execute a query asynchronously, you call the **OpenResultset** method with the **rdAsyncEnable** flag. Then you enable the timer. At each interval when the timer fires, it checks the status of the current query. If the query is completed, it calls a routine to handle the results. To keep this simple, we are using a technique that allows only one query at a time. Alternatively, there are many techniques that use arrays or round-robins to handle multiple queries at the same time.

```
Sub ExecuteAsyncQuery(connWork as rdoConnection, ByVal sQry as String, _
                      ByVal iType as Integer, ByVal iLockType as Integer
    'rsWorker is an rdoResultSet object defined public
    Set rsWorker = connWork.OpenResultSet(sQry, iType, iLockType, rdAsyncEnable)
     tmStatus.Enabled = True
End Sub

Sub tmStatus_Timer()
tmStatus.Enabled = False        'So no re-entrancy
If rsWorker.StillExecuting Then
    tmStatus.Enabled = True        'It's still working...
Else
    ProcessResults
End If
End Sub
```

To cancel this query, you might use code that looks like this:

```
Sub btCancel_Click()
tmStatus.Enabled = False
If rsWorker.StillExecuting Then
    rsWorker.Cancel
    Set rsWorker = Nothing
Else
    If Not fAlreadyProcessing Then ProcessResults
End if
End Sub
```

There are many situations where using asynchronous queries is preferable. Basically, you want to use asynchronous queries when you have a long process that needs to be completed, but you don't want to tie up an end-user computer to complete it.

Microsoft SQL Server tasks are an alternative to asynchronous queries. *Tasks* are commands that the Microsoft SQL Server Executive launches once or at desired intervals. Because these tasks are run on the server, they are asynchronous to a client requesting their execution. They can also make use of logging and alert mechanisms, making this a very powerful and rich alternative to asynchronous queries. For more information on tasks and alerts, see my Client/Server Solutions: Leveraging Microsoft SQL Server Services in a Transaction Processing Environment technical article.

When using RDO or DAO 3.1 for your data access mechanism, it becomes possible to have queries that produce more than one set of results in a single **resultset** or **recordset** object. When a query returns more than one set of results, the **resultset** or **recordset** object initially presents only the first set. To gain access to the successive sets, you must call the **MoreResults** method in RDO or the **NextRecordset** method in DAO 3.1. By doing this, you flush the contents of the current set of results and move to the next one. You may not reverse this operation. Here is a sample using a query that returns the content that will fill several list box controls; again, I've chosen to implement this in RDO.

```
Sub Prepare()
Dim rsWork as ResultSet 'Use the public conn
Dim iIndex as Integer

iIndex = 0
Set rsWork = connMain.OpenResultSet(sLstQry)
If rsWork.BOF and rs.EOF Then
    'No data returned, do error handling
Else
    Do Until rsWork.EOF
        lstData(iIndex).AddItem Trim$(rsWork("Description"))
        lstData(iIndex).ItemData(lstData(iIndex).NewIndex) = CLng(rsWork("PKId"))
        rsWork.MoveNext
    Loop
    Do While rsWork.MoreResults
        iIndex = iIndex + 1
        Do Until rsWork.EOF
            lstData(iIndex).AddItem Trim$(rsWork("Description"))
            lstData(iIndex).ItemData(lstData(iIndex).NewIndex) = _
                CLng(rsWork("PKId"))
            rsWork.MoveNext
        Loop
    Loop
End if
End Sub
```

Why Use Multiple Result Sets?

Developing client/server applications today means considering efficient use of distributed resources. Going over wide-area networks is much more costly than pinging the server two floors down. Using multiple result sets can create more options in the way that transactions are packaged. For example, the task of retrieving data to populate a form might involve data from several tables or sources. Packing the requests into a single statement or stored procedure and allowing the server to send back multiple results eliminates the need for the client to handle multiple sessions with the server. It turns this type of communication:
{ewc mvimg, mvimage,!illust.bmp}

Into this:
{ewc mvimg, mvimage,!illust.bmp}

The latter organization is much more efficient when the cost of network requests is high.

You can use this type of optimization, for example, in order to:

® Send all the lookup data for a form back to the client in one shot.

® Provide information of different sizes, such as a customer information record as well as order history, issues tracking, and the like, in the same request.

Ultimately, RDO and the new features in the release of DAO 3.1 will have a significant impact on the performance of client/server applications. However, full realization of these benefits will require a proper use of these new features and possibly an adjustment to the way we design transaction models in the future.

In this exercise, you will enable Anonymous logon, but you will set permissions on the transcript.asp page to only allow users with valid Windows NT accounts to view that page.

Enable Anonymous logon

1. Run the Internet Service Manager.
2. Right-click the computer name for the WWW service, and click **Service Properties**.
3. On the **Service** tab, select the **Allow Anonymous** check box in the Password Authentication group, and then click OK.

   Both **Allow Anonymous** and **Basic** should now be selected.

Set permissions on transcript.asp

1. Run Windows NT Explorer.
2. Go to the StateU folder in the installation folder for your Web server, right-click the file transcript.asp, and click **Properties**.

   If you accepted the default options when you installed Microsoft Internet Information Server (IIS), the file transcript.asp will be in the folder C:\InetPub\wwwroot\StateU.
3. On the **Security** tab, click **Permissions**.
4. In the **File Permissions** dialog box, click the Internet Guest Account for your Web server, set the **Type of Access** to **No Access**, and then click OK.

   The Internet Guest Account is typically IUSR_*computername*.

   For information about controlling access to a file, see Setting NTFS Permissions in this chapter.

Test

1. Exit Microsoft Internet Explorer and then restart it.
2. Go to the State University home page, default.htm.

   You are logged in with Anonymous authentication and will not be prompted for a user name or password.
3. Go to the page transcript.asp.

   The Anonymous account is not allowed access to this page, so you are prompted for a valid Windows NT account or password.
4. Enter a valid Windows NT user name and password.

   The transcript page is displayed.

In this exercise, you will create the page for the links frame of the State University Web site.

Create the links.htm page

1. In the StateU Web project, create a new HTML page named links.htm.
2. Open links.htm with the FrontPage Editor.
3. Set the background color to white.

   For information about setting the background color and image, see Setting Page Properties.
4. Insert a two-column, nine-row table in the HTML page.

   The following illustration shows how the table should look.

   {ewc MVIMG, MVIMAGE,!W02G065.bmp}

   a. Set the border size of the table to zero.

   b. Insert the file images/bullet.gif into the first column of the table.

   c. Enter text in the table cells, as shown in the table illustration.

   d. Set the cell properties of the three headings to **Header Cells**, and make the headings span two columns.

   For information about creating tables, see Creating Tables.
5. Create hyperlinks from the table to other pages in the State University Web site, as shown in the following table.

Text	Hyperlink
General Information	home.htm
Provide Feedback	feedback.htm
Course Catalog	classlist.asp
Course Descriptions	class_descriptions.asp
Get Transcript	transcript.asp
Register for Course	classview.htm

6. After the table, insert the image of the State University mascot (images/mascot.gif), and make it a hyperlink to the page mascot.htm.
7. Save your changes to links.htm.
8. View the page in Microsoft Internet Explorer with the FrontPage Editor by clicking **Preview in Browser** on the **File** menu.
9. View the completed home page of the State University Web site by going to default.htm in Microsoft Internet Explorer.

   Click the mascot image in the links frame. The page mascot.htm displays in the lower-left frame, instead of displaying in the frame on the right.

Fix the links in links.htm

Edit the file links.htm so the hyperlinks will appear in the frame on the right.

1. In the FrontPage Editor, edit the file links.htm.
2. On the **File** menu, click **Page Properties**.
3. On the **General** tab, type **main** in the **Default target frame** box.
4. Save your changes to links.htm, and then close FrontPage.
5. View default.htm in the Visual InterDev InfoViewer, and test the link to mascot.htm.

Scalable Systems: SMP Hardware Systems and Clusters

Today, downsizing computer systems to client/server computing is a common phenomenon. Yet many successful companies face the opposite problem: they are upsizing their client/server applications. With every Internet and intranet user a potential client, applications face huge user and transaction loads. With the price of disk storage at $100/GB, application systems are encouraged to store huge online databases.

Scalable systems solve the upsizing problem by giving the designer a way to grow the network, servers, database, and applications by just adding more hardware. Scalable computer systems can grow an application's client base, data base, and throughput without application reprogramming. The scaled-up server should be as easy to manage as the smaller system — at least on a per-user basis.

Traditionally, most scale-up has been done thorough *symmetric multiprocessing* (SMP): adding more processors, memory, disks, and network cards to a single server. Several vendors have shown that SMP servers can offer a ten fold scale-up over uniprocessor systems on commercial workloads.

However, at some point a single node hits a bottleneck. This results in diminishing returns or a need for prohibitively expensive hardware. To grow much beyond a factor of 10, application designers have gravitated to a *cluster* architecture in which the workload and database are partitioned among an array of networks, processors, memories, and database systems. All the truly large systems are built this way. The IBM MVS Sysplex, the Digital Equipment Corporation (DEC) VMScluster, the Teradata DBC 1024, and the Tandem Himalaya series are all clustered systems. Ideally, this partitioning is transparent to the clients and to the application. The cluster is programmed and managed as a single system, but it is in fact an array of nodes.

Cluster technology dovetails with and leverages distributed system technologies such as replication, remote procedure call, distributed systems management, distributed security, and distributed transactions. The cluster approach has two advantages over increasingly larger SMP systems: (1) clusters can be built from commodity components and can grow in small increments, and (2) the relative independence of cluster nodes allows a natural fail-over and high availability design.

Microsoft SQL Server and Windows NT Server: Scalability on Both SMP and Clusters

Microsoft Windows NT Server and Microsoft SQL Server support *both* SMP and cluster architectures. SQL Server delivers impressive peak performance for both client-server and data warehouse applications. Microsoft SQL Server and Microsoft Windows NT Server run on two-way to 32-way SMP hardware systems, although the most common SMP hardware systems are two-way and four-way. Microsoft SQL Server has demonstrated excellent SMP scalability, both on audited benchmarks and in real applications.

Today, a single four-processor node can support up to 5,000 concurrent users accessing a Microsoft SQL Server, storing billions of records on a 200-GB disk array (Table 1). These servers are capable of processing more than 10 million business transactions per day. The combination of faster processors, more processors, and software improvements could more than double the rated performance of Microsoft SQL Server on SMP hardware systems. Microsoft plans to publish audited benchmarks in 1996 that demonstrate Microsoft SQL Server scalability to eight-processor SMP hardware systems. Larger SMP hardware systems will be supported as the technology matures.

Table 1. Microsoft SQL Server Scalability Today (April 1996)

Technologies	Active Users	Throughput	DB Size
SMP, fail-over, parallel query, distributed transactions, SQL Enterprise Manager	5,000 per SMP	3,600 transactions per minute (tpm) 10 million transactions per day (tpd)	200 GB

Microsoft SQL Server and Windows NT Server Scalability

Microsoft plans to evolve Windows NT Server to support large processor and disk clusters. These plans are outlined in the paper "Microsoft Windows NT Server Cluster Strategy," in the MSDN Library. Today, Windows NT Server offers many clusterwide services, including: security, naming, performance monitor, software asset

management (Systems Management Server), transactions (Distributed Transaction Coordinator), distributed objects (Distributed Component Object Model), node and resource fail-over, and automatic Internet Protocol (IP) address assignment (Dynamic Host Interface Protocol). Companies such as Compaq, Digital Equipment Corporation, Hewlett-Packard, Intel, NCR, Tandem, and SGI are building on Windows NT Server and are working with Microsoft to make Windows NT Server clusters even more powerful. Many additional services are planned, including an industry standard cluster application programming interface.

Microsoft SQL Server Version 6.5 Clustering and Fail-Over

Microsoft SQL Server is an early user of clusters for fail-over support. Microsoft SQL Server 6.5, in cooperation with various hardware vendors, supports high-availability databases via fail-over from one node to another. Microsoft SQL Server also adds support for OLE Transactions through the Microsoft Distributed Transaction Coordinator (MS DTC). OLE Transactions allow Microsoft SQL Server databases to be partitioned among multiple Windows NT Servers. MS DTC automatically manages the work of transactions that span these nodes. In addition, the Microsoft SQL Enterprise Manager allows an operator to monitor and manage multiple SQL Servers from a single console.

Goals for the Future

Fail-over and OLE Transactions are the most recent steps toward Microsoft's vision of superservers built from clusters of Windows NT Servers. Windows NT Server clusters will provide *modular growth,* allowing customers to buy only what they need and to grow the system by adding processing, storage, and network modules to the server as demand rises. Microsoft aims to make it easy to build and manage these superservers — indeed, we want to bring the ease of plug-and-play to enterprise clusters. We hope to automate much of the work of configuring and managing a cluster. The clusters will offer high-availability services by automatically reconfiguring to mask system failures.

As seen in Table 2, Microsoft's goal for 1998 is to support 16-node Windows NT Server clusters, where each node is a four-way SMP with one hundred discs (800 GB). In 1998, Microsoft SQL Server will offer transparent access to data partitioned in this cluster. This 16-node cluster is potentially a 10-terabyte database. Such a system should be big enough for most application problems. Whether current price trends continue, in 1998 such a cluster could be built from commodity components for about 1 million dollars.

Table 2. Microsoft SQL Server Scalability Goals

By year end	Technologies	Active Users	Throughput	DB Size
1996	SMP, fail-over, parallel input/output (I/O), distributed transactions, SQL Enterprise Manager	5,000–7,000 per SMP	5,000–7,000 tpm 5 M tpd	500 GB–1 terabyte
1997	SMP, fail-over Distributed Server management	10,000 per SMP	10,000 tpm	1–5 terabytes
1998	16-node clusters of *N*-way SMP machines Transparent partitioning Parallel query 64-bit support	15,000 per SMP with clusters, up to 200,000	18,000 tpm	10 terabytes/ cluster

Once the clustering and transparent partitioning are in place, Microsoft SQL Server will implement automatic parallel access to the database.

Kinds of Growth

As organizations grow and acquire more and more data, they must deal with increased transaction workloads and larger databases. Each major increment presents new challenges.

Scalability covers several kinds of growth, including:

- **Growing user population and network.** Whether the user population doubles, the network and database workload probably do also.
- **Growing database size.** With databases commonly reaching hundreds of gigabytes, operations such as backup, restore, and load all can become bottlenecks.
- **Growing transaction complexity.** Application designers are building more intelligence into applications, relieving users of tedious tasks. Increasingly, applications are used for data mining and data analysis.
- **Growing applications.** As applications become easier and less expensive to build, organizations are using computers in new ways. These new applications increase the load on existing databases and servers.
- **Growing numbers of servers.** Clustered and distributed applications involve many nodes. Scalable systems allow a large number of nodes to be easily managed from a single location.

Scale-up, Speed-up, and Scale-out

An ideal system improves linearly: that is, whether you double the number of processors and disks, throughput doubles (linear scale-up) or response time is cut in half (linear speed-up). Linear scaling is rarely achieved in practice because it requires that all aspects of the system be perfectly scalable. The overall architecture includes hardware, operating system, database, network software, and the application software.

It is tempting to simplify scalability to a single metric such as the number of processors a system can support. It may seem obvious but many database applications are very I/O-intensive; adding central processing units (CPUs) to an I/O-bound system will not make it faster. Microsoft SQL Server running on today's typical four-processor servers can achieve performance comparable with systems running on hardware with 20 processors! Is the second system more scalable because it requires 20 CPUs to achieve the same throughput? Obviously not. Table 3 shows some benchmarks using many processors. It shows that a four-processor Compaq running Windows NT Server and Microsoft SQL Server outperforms many eight-processor and some 20-processor systems.

Table 3. Windows NT Server and Microsoft SQL Server vs. SMP UNIX Solutions on the TPC-C Benchmark

Database	Hardware	CPUs	tpmC	$/tpmC	System Cost
Microsoft SQL Server 6.5	Compaq Proliant 4500/133	4	3643	$148	$537,508
DB2 for Solaris V2	Sun SPARC Server1000E	8	3256	$199	$646,820
Informix Online 7.1	Sun SPARC Center 2000E	20	3534	$495	$1,751,004
Oracle 7.3	IBM RS6000 / j30	8	3631	$ 289	$1,049,656

The best metric is how well the system scales for your application. That is impossible to know in advance. But, you can estimate scalability by examining standard benchmarks and by looking at applications in related industries.

Scalability and Manageability

Performance is only a small part of the cost of ownership. System management and operation is often more expensive than the hardware and software. Larger systems can be difficult to design, program, administer, and operate. The following issues are important for small systems, but they become *very* important when the systems scale up.

- **Manageability.** A scalable system should automate periodic and repetitive system management tasks. Operators should only be involved in exception situations that require administrative decisions and even those tasks should have a graphical interface assisted by wizards.
- **Continuous availability.** The cost of downtime increases with system size. Unfortunately, larger systems have more components and so are more likely to break. Scalable systems must deliver continuous 24-hour, seven day-a-week availability. They must allow online operations (backup, recovery, reorganization, and upgrades) and must mask hardware and software failures with automatic fail-over to another server.
- **Price/Performance of commodity hardware.** Any solution must use cost-effective hardware platforms. Approaches based on specialized architectures are increasingly uneconomic.

Customer experience, magazine reviews, and independent studies have all shown that Windows NT Server and Microsoft SQL Server can dramatically help reduce support and management costs when compared with non-Microsoft systems.

This section includes the following topics:

Technology Trends Encourage Building Scalable Systems

Technology advances and the wide adoption of computing have had some surprising results. Today, commodity components are the fastest and most reliable computers, networks, and storage devices. The intense competition among microprocessor vendors has created incredibly fast processing chips. Indeed, the most powerful supercomputers are built from such chips. Conventional water-cooled mainframes are moving to this higher-speed technology.

Commodity workstations and servers rival and often outstrip the performance of mainframes. The pace of change in this commodity market is so rapid that the "low end" is years ahead of the installed base of conventional mainframe and minicomputer architectures. Figure 1 shows the astonishing increase in microprocessor speed over time.

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. Intel microprocessor speed vs. time

Commodity computer interconnects are also making extraordinary progress. Ethernet has graduated to 100 megabits per second (Mbps). Switched 100 Mbps Ethernet gives a hundred fold speed-up in local networks at commodity prices. Asynchronous Transfer Mode (ATM) hardware will soon be ubiquitous for both local and wide area networks. ATM offers a tenfold increase over switched Ethernet.

The entire computer industry uses the same basic 16-MB dynamic RAM (DRAM) chips that go into PCs. Memory prices for commodity machines are often three to ten times less than the price of the same memory for a proprietary machine.

The highest performance and most reliable disks are 3.5" small computer system interface (SCSI) discs. They have doubled in capacity every year and are rated at a 50-year mean time to hardware failure. Today, the 4-gigabyte disk is standard. Next year, it will be replaced by the 8-gigabyte drive for approximately the same price. Figure 2 shows that in 1995 one thousand dollars bought 4 GB of disks. This is a thousandfold improvement over the disks of 15 years ago. These low prices explain why typical servers are configured with 10 GB to 100 GB of disk capacity today. Such disks cost $2,000 to $20,000.

{ewc mvimg, mvimage,!illust.bmp}
Figure 2. Megabytes per $1000 vs. time

Computer Architecture Choices: SMP Hardware Systems and Clusters

The proliferation of many processors, disks, and networks poses an architectural challenge. What hardware architecture best exploits these commodity components? No single architecture has emerged as a winner, but there is broad agreement that three generic architectures can provide scalability: shared memory, shared disk, and shared nothing. Windows NT Server supports all these architectures and will evolve in parallel as these architectures evolve.

SMP grows a server by adding multiple processors to a single shared memory. The system grows by adding memory, disks, network interfaces, and processors. SMP is the most popular way to scale beyond a single processor. The SMP software model, often called the shared memory model, runs a single copy of the operating system with application processes running as if they were on a single processor system. Microsoft Windows NT Server and Microsoft SQL Server are designed to scale well on SMP systems. They can scale to 32 nodes for some applications, but the practical limits for general purpose use today are:

® eight processors.

® 4 gigabytes of main memory (2 gigabytes per application).

® 125 disk drives configured as 25 logical disks holding 500 GB with 4-GB disks, 1 terabyte with 8-GB disks.

® 5,000 active clients accessing a SQL server.

These are the maximum sizes we have seen. Typical large servers are half this size or less. With time, Microsoft SQL Server, Windows NT Server, and hardware technology will improve to support even larger configurations.

SMP systems are relatively easy to program. They also leverage the benefits of industry standard software and hardware components.

Today, SMP is by far the most popular parallel hardware architecture. SMP servers based on industry-standard Intel and Alpha AXP microprocessors deliver astonishing performance and price/performance for database platforms. Intel markets a four-way SMP board based on the Pentium Pro (P6) processor. This SMP board is being incorporated in servers from almost every hardware vendor. It appears that the 4xP6 SMP servers will be the workhorse of client-server computing for the next few years.

As microprocessor speeds increase, SMP systems become increasingly expensive to build. Today, there are modest price steps as a customer needs to scale from one processor to four processors. Going from four to eight processors in an SMP is comparatively expensive. Beyond eight processors, prices rise very steeply and the returns diminish.

For example, as can be seen in Table 4, Compaq can deliver astonishing performance on a standard, widely available four-processor SMP machine. It is designed using commodity hardware and delivers very cost effective database support. A 16-processor machine, designed around that same 133-MHz Pentium chip, delivers only 50 percent more performance, at over four times the cost.

Table 4. Diminishing Returns on SMP Performance

Database	Hardware	CPUs	tpmC	$/ tpmC	System Cost
Microsoft SQL Server 6.5	Compaq Proliant 4500/133	4 Pentium 133 MHz	3643	$148	$537,508
Oracle 7 v 7.3.2	NCR WorldMark 5100S c/s	16 Pentium 133 MHz	5607	$394	$2,206,688

At the software level, multiple processors concurrently accessing shared resources must be serialized. This serialization limits the practical scalability for individual applications in shared memory SMP systems. These software bottlenecks in operating systems, database systems, and applications are as significant as the hardware limitations.

Nonetheless, SMP systems are the most common form of scalability. They will be the dominant form of server for many years to come. The appearance of Intel Pentium Pro, DEC Alpha, IBM PowerPC, and SGI MIPS SMP hardware systems gives very powerful and inexpensive SMP nodes.

Microsoft provides excellent support for these commodity SMP hardware systems today and continues to invest in this technology.

A cluster is a set of loosely coupled, independent computer systems that behave as a single system. The nodes of a cluster may be single-processor systems, or they can be SMP hardware systems. The nodes may be connected by a commodity network or by a proprietary very highspeed communications bus. The computers in a cluster cooperate so that clients see the cluster as if it were a single very high performance, highly reliable server. Because the cluster is modular, it can be scaled incrementally and at low cost by adding servers, disks, and networking.

Windows NT Server clusters provide *horizontal growth*, or *scale-out*, allowing customers to add processing, storage, and network services to a pre-existing configuration. Figure 3 shows the growth from a one-node cluster to a six-node cluster by adding one node at a time. A cluster can also be grown using superserver SMP nodes, but usually it is built with commodity components and commodity interconnects. Two interconnects are used for fault tolerance.

{ewc mvimg, mvimage,!illust.bmp}
Figure 3. Cluster growth, one node at a time

Cluster architectures allow you to scale up to problems larger than a single SMP node. Microsoft believes that the cluster is the most economical way to achieve scalability beyond eight processors. When demand exceeds the ability of commodity SMP nodes, or when fault tolerance demands a second fail-over server, it is very attractive to use multiple nodes to form a cluster.

Shared-disk and Shared-nothing Clusters

Figure 4 shows the two basic software models for clusters: shared-disk and shared-nothing. In a *shared-disk cluster*, all processors have direct access to all disks (and data), but they do not share main memory. An extra layer of software, called a distributed cache or lock manager, is required to globally manage cache concurrency among processors. The Digital VMS cluster and the Oracle Parallel Query Option are the most common examples of a shared-disk parallel database architecture. Since access to data is serialized by the lock or cache manager, the shared-disk cluster has some of the same scalability limitations as shared-memory SMP systems.

{ewc mvimg, mvimage,!illust.bmp}
Figure 4. Shared-disk and shared-nothing clusters

The *shared-nothing cluster* architecture avoids the exotic interconnects and packaging of a scalable shared-disk cluster by connecting each disk to one or two servers. In a shared-nothing cluster, each node has its own memory and disk storage. Nodes communicate with one another by exchanging messages across a shared interconnect. Each node in a shared-nothing cluster is a freestanding computer, with its own resources and operating system. Each node is a unit of service and availability. Each node owns and provides services for some disks, tapes, network links, database partitions or other resources. In case of node failure, the disks of one node may fail over to an adjacent node — but at any instant, only one node is accessing the disks.

Proprietary shared-nothing cluster solutions have been available for several years from Tandem, Teradata, and IBM Sysplex. These solutions all depend on exotic hardware and software, and therefore are expensive.

Microsoft SQL Server and Windows NT Server clusters will bring scalability and fault tolerance to the commodity marketplace. Microsoft is building clustering technology directly into the Windows NT Server operating system. This technology will work well with commodity servers and interconnects, and it will be able to leverage special hardware accelerators from vendors such as Compaq, Digital, and Tandem. Microsoft BackOffice products, such as Microsoft SQL Server, Internet Information Server, and Exchange, will be enhanced to take advantage of this clustering support. Many third-party products are also being ported to this architecture. Later, this whitepaper discusses Microsoft's strategy to support clustering in both Windows NT Server and Microsoft SQL Server.

This section includes the following topics:

® Kinds of Parallelism

® Windows NT Server: Parallel Processing and Parallel I/O

Software and applications running on an SMP or a cluster must use parallelism to benefit from the many disks and processors that these scalable architectures provide. Figure 5 depicts the two generic kinds of parallelism.

{ewc mvimg, mvimage,!illust.bmp}
Figure 5. Pipeline and partition parallelism

Pipeline parallelism breaks a big task into a sequence of steps. Each step of the sequence is done in parallel. The result of one step is passed to the next step. An industrial assembly line is a good example of pipeline parallelism.

Partition parallelism breaks a big task into many little independent tasks that can be done in parallel. Having multiple harvesters collect grain from several fields is a typical example of partition parallelism.

Microsoft SQL Server provides a good example of both partition and pipeline parallelism when reading disks. By striping data across many disks, each disk can deliver a part of a large data read — this is partition parallelism. Microsoft SQL Server exploits three-deep pipeline parallelism by having one task issue pre-fetch reads to the disks, while another Microsoft SQL Server task processes the data and passes it back to the application task.

Windows NT Server was designed for SMP hardware systems. Windows NT Server is a fully threaded operating system that provides a rich set of synchronization primitives. The shrink-wrapped version of Windows NT Server is enabled for four-way SMP. Larger SMP hardware systems are not yet a commodity product, so the operating system is generally packaged with the hardware. Companies such as Digital, Intergraph, NCR, Sequent, and SGI, who build 8-way and 32-way Windows NT Server multiprocessors, modify Windows NT Server for their hardware and ship it with the hardware package.

Windows NT Server also has built-in disk partitioning. It allows logical volumes to be spread across multiple physical volumes. Each logical volume can be a *redundant array of inexpensive disks* (RAID) or it can be an individual disk. Windows NT Server has software support for RAID 0 (striping a logical volume across multiple physical volumes), RAID 1 (disk mirroring or shadowing in which data is replicated on multiple disks), and RAID 5 (a fault-tolerant storage strategy in which disks are organized so that one stores parity for data on the others). RAID 5 is more space efficient than RAID 1. This software support lets Windows NT Server customers build reliable storage with commodity controllers and disks. RAID also offers partition parallelism because it spreads the data among multiple disks that can be read in parallel.

Windows NT Server can read individual commodity SCSI disks at their rated speed of 8 MB/second and can drive SCSI controllers at speeds of 18 MB/second. By using a Peripheral Component Interconnect-based (PCI-based) machine and by striping data across four SCSI controllers, a single-commodity Pentium processor running Windows NT Server has been clocked reading a disk array at 60 MB/s. This means that a single Pentium processor can read or write data at 200 gigabytes per hour. These rates are important for data intensive operations such as data loading, backup, data indexing, and data mining.

These are impressive numbers for any computer, but they are especially impressive when using commodity single-processor systems. By using multiple processors and multiple PCI busses, SMP systems can read and write data at even higher speeds and still have power left over to analyze and process the data.

Microsoft SQL Server is designed to make full use of the Windows NT SMP capabilities: threads and the Windows NT Server scheduler. Each Windows NT Server node typically has a single Microsoft SQL Server address space managing all the SQL databases at that node. SQL Server runs as a main address space with several pools of threads. Some threads are dedicated to housekeeping tasks such as logging, buffer pool management, servicing operations requests, and monitoring the system. A second larger pool of threads performs user requests. These threads execute stored procedures or SQL statements requested by the clients.

Microsoft SQL Server is typically used in a client/server environment where clients on other machines attach to the server and either issue SQL requests, or more typically, issue stored procedures usually written in the Transact-SQL language. Clients may also be co-located at the server node. Microsoft SQL Server uses a built-in transaction processing (TP) monitor facility, Open Data Services, to support large numbers of clients. In practice, this has scaled to 5,000 concurrent clients. Beyond that size, it makes sense to either partition the application into a cluster of nodes or use a TP monitor to connect clients to SQL Server. All common TP monitors, such as IBM CICS, Transarc Encina, Novell Tuxedo, and Top End from AT&T Global Information Solutions, have been ported to Windows NT Server and interface with Microsoft SQL Server.

Microsoft SQL Server has devoted considerable effort to achieving near-linear speed-up on SMP machines. Many benchmarks have demonstrated this. Two notable examples are the DEC Alpha TPC-C Benchmark and the Intergraph benchmark.

Benchmarks Showing Microsoft SQL Server SMP Scalability on TPC-C

There is no universally accepted way to measure scalability. However, useful information can be gained from Transaction Processing Performance Council (TPC) benchmarks. The TPC is a nonprofit organization that defines industry standard transaction processing and database benchmarks. Members of the council today include all major database vendors and suppliers of server hardware systems. They have defined a series of benchmarks: TPC-A, TPC-B, TPC-C, and TPC-D.

TPC-C is the current industry standard benchmark for measuring the performance and scalability of online transaction processing (OLTP) systems. It exercises a broad cross section of database functionality including inquiry, update, and queued mini-batch transactions. The specification is strict in critical areas such as database transparency and transaction isolation. Many consider TPC-C to be a good indicator of real-world OLTP system performance. The benchmark results are audited by independent auditors and a full disclosure report is filed with the TPC. These reports are a gold mine of information about how easy it is to use the various systems and how much the systems cost.

Several vendors have reported scale-up results that demonstrate scalability from one CPU to a multiple-CPU SMP. Figure 6 shows the reported SMP scalability data as of March 1, 1996. At present, IBM DB2, ORACLE, and SYBASE have not reported scale-up numbers that give a series from one CPU up to multiple-CPU SMP hardware systems. Only Informix and Microsoft SQL Server have reported such results. The chart shows two interesting things. First, the DEC Alpha and Microsoft SQL Server absolute performance on small SMP hardware systems is excellent. Indeed, a three-processor DEC Alpha has twice the throughput of a six-processor NEC MIPS platform. Microsoft SQL Server also has more throughput than Informix running on an eight-processor 166-MHz PowerPC from Bull. One must go to an eight-processor 200-MHz SNI MIPS system running Informix to beat the three-processor DEC Alpha number in these scale-up measurements. It also shows that a three-processor DEC Alpha running Microsoft SQL Server 6.5 and Windows NT Server version 3.51 gives 92 percent scalability (that is, three processors give 2.76 times the single-processor throughput). The SNI/Informix scale-up gives less than 55 percent at eight processors and less than 45 percent at 16 processors. These audited results clearly show that Microsoft SQL Server 6.5 and Windows NT Server 3.51 deliver excellent SMP scalability for up to four nodes.

{ewc mvimg, mvimage,!illust.bmp}
Figure 6. TPC-C throughput vs. number of CPUs

In laboratory tests, Intergraph Corporation demonstrated the scalability of Microsoft SQL Server 6.0 running on Intergraph's high-end InterServe Multiprocessor 6 (ISMP6) server (six 100-MHz Pentium processors with 1-MB secondary caches). Compared to Microsoft SQL Server 4.21a, SQL Server 6.0 achieved greater than 80 percent scalability on the debit/credit transaction processing benchmark (Figure 7).

{ewc mvimg, mvimage,!illust.bmp}
Figure 7. Scalability: Microsoft SQL Server 6.0 vs. Microsoft SQL Server 4.21a

The Intergraph study concludes that "Microsoft SQL Server 6.0 gained much more performance from the additional processors than Microsoft SQL Server 4.21a, even though both were running the same benchmark, on the same hardware, and on the same operating system. The differences are especially noticeable as the number of processors increases over four, since the effect is geometric. This result supports Microsoft's claim that vendors must optimize their applications to take full advantage of the underlying operating system. Obviously, the application has a considerable affect on scalability, and Microsoft SQL Server 6.0 on Windows NT Server 3.51 scales well beyond 4 processors." Microsoft SQL Server 6.0 was the 1995 release. It has been replaced by Microsoft SQL Server 6.5 running on Windows NT Server 4.0. As the newest TPC-C results show, both Microsoft SQL Server 6.5 and Windows NT Server 4.0 have even better SMP scalability.

Figure 8 gives a sense of how much Microsoft SQL Server and Windows NT Server have improved over the last three years on commodity processors. It shows that on the debit/credit benchmark throughput increased from 94 ticks per second (tps) to over 1,400 tps on a single processor — this represents a 120 percent annual growth rate for each of the last three years. In part, this 15-fold improvement comes from hardware advances: the processor speed-ups, larger memory, and improved disks. A major component of these improvements came from Microsoft SQL Server and Windows NT Server. New SMP and database algorithms in Windows NT Server and Microsoft SQL Server exploit these hardware advances. The result is a huge improvement in just three years. We expect these software and hardware improvements to continue for several more years.

{ewc mvimg, mvimage,!illust.bmp}
Figure 8. Debit/Credit throughput of Microsoft SQL Server and Windows NT

A single Microsoft SQL Server and Windows NT Server can support thousands of users accessing a database containing billions of records. Such systems are capable of supporting a user community exceeding 20,000 or a much larger community of Internet users who are occasionally connected to the server. Just to give a sense of scale, the largest banks have less than 10,000 tellers and the largest telemarketing organizations have less than 10,000 active agents. So, in theory, these systems could support the traditional front office of huge corporations.

Every SMP system has a maximum size: a maximum number of nodes, memory, disks, and network interfaces that can be attached to the system. To go beyond these limits, the software architecture must support a cluster or network of computers cooperating to perform the common task.

All the largest computer systems are structured as clusters. The largest known database, the Wal-mart system, runs on a NCR/Teradata cluster of 500 Intel processors accessing approximately 1,500 disks. Other large databases are supported by IBM MVS/Sysplex systems, Tandem Himalaya systems, or Digital VMScluster systems.

Clusters work by partitioning the database and application among several different nodes (computers). Each node stores a part of the database and each performs a part of the workload. The cluster grows by adding nodes and redistributing the storage and work to the new nodes. This growth and redistribution should be automatic and require no changes to application programs.

Partitioned data is used in centralized systems to scale-up capacity or bandwidth. When one disk is too small for a database, the database system partitions it over many disks. This same technique is used if traffic on a file or database exceeds the capacity of one disk. This partitioning is *transparent* to the application. The application accesses all data on all the disks as though it were on a single disk. Similarly, clients are partitioned among networks to scale up network bandwidth. Clusters generalize this partitioning to spread the computation among cluster nodes. The key challenge is to make this partitioning transparent to the customer so that a cluster is as easy to manage and program as a single system.

Clusters: Growing Beyond SMP Limits

Clusters have several advantages. The most obvious is that they can grow to huge sizes. By building clusters from SMP nodes, one can scale up an application by a factor of ten or a hundred. Indeed, Digital, Tandem, and Teradata have many 100-node clusters in production today.

Clusters of Commodity Components are Economical

Clusters can be built with high-volume components that are relatively inexpensive. Relatively few high-end SMP machines are sold — relatively few customers will buy a 20-way SMP. So, 20-way SMP hardware systems are expensive because the engineering cost is amortized over relatively few units. Inexpensive clusters can be built with commodity four-way SMP hardware systems. This means they have superior price/performance. In addition, because they are built from commodity components, commodity clusters can grow in smaller increments. You can add disks or nodes or network cards as needed rather than having to buy a huge new box each time you grow.

Availability via Fault Tolerance

By partitioning data and work, clusters provide firewalls that limit the scope of failures. When a node fails, the other cluster nodes continue to deliver service. By replicating data or by allowing storage and network devices to fail over to surviving nodes, the cluster can provide uninterrupted service.

In summary, clusters have substantial advantages. Microsoft believes that commodity SMP hardware systems will be the building blocks for clusters. Availability, scalability, and economics will demand that large systems be built as arrays of these commodity components. SMP hardware systems will provide vertical growth of nodes from single-processor systems to large SMP hardware systems. Clusters will provide horizontal growth by combining these SMP hardware systems into Windows NT Server clusters.

This section includes the following topics:

Distributed systems techniques are the key to building transparency into clusters. By structuring applications and systems as modules that interact via remote procedure calls, applications become more modular, and they can be distributed among nodes of the cluster. The client calls upon a service by name. The procedure call either invokes the service locally or uses a remote procedure call if the service is remote.

Microsoft has invested heavily in structuring its software as components that interact via remote procedure calls. The resulting infrastructure is variously called OLE, component object model (COM), and distributed COM (DCOM). Many aspects of COM are in place today and more are coming soon. In particular, with Windows NT Server 4.0 and Microsoft SQL Server 6.5, Microsoft is delivering the following:

® DCOM, supporting distributed invocation of objects in a network based on remote procedure calls.

® OLE Transactions (also called Distributed Transaction Coordinator) that allow applications to do work at many Microsoft SQL Server database partitions and automatically (transparently) get the ACID (atomicity, consistency, isolation, and durability) distributed transaction semantics.

® OLE DB, which allows Microsoft SQL Server (and any other data integrator) to access data from any data source supporting the OLE DB interface. (Planned for shipping in early 1997, an SDK for OLE DB became available in late 1996.)

Windows NT Server already supports these features along with many other cluster facilities, including cluster security (domains), cluster software management (SMS), cluster naming (Distributed Name Service), and cluster performance monitor (Perfmon). Microsoft SQL Server complements these facilities with management tools built into the Microsoft SQL Server Enterprise Manager that allow it to manage and monitor an array of Microsoft SQL Servers.

In 1997, Windows NT Server and SQL Server will extend many of these interfaces and will incorporate facilities to manage and balance application servers across a cluster of 16 nodes. The cluster will have a common console and a simple management model for all the components. Microsoft's goal for Windows NT Server and Microsoft SQL Server is to make the cluster as easy to manage and use as a single large system. Clusters will have the added benefit of fault tolerance, masking many system faults and providing highly available services.

Microsoft SQL Server 6.5 offers fault tolerance and high availability by providing fail-over from one server to another when the first server fails or needs to be taken out of service for maintenance (Figure 9). The fail-over mechanism works as follows: Two Windows NT Server servers can be configured as dual servers. The pair must have shared access to a SQL Server database stored on shared disks. One of the Microsoft SQL Servers is the primary server. It accepts requests from clients and both reads and writes the database. The other (fail-over) Microsoft SQL Server node may be providing access to other databases, but it is not accessing the database being managed by the primary. If the primary Microsoft SQL Server node fails, the Windows NT Server at the fail-over node takes ownership of the disks holding the database. It then informs the fail-over Microsoft SQL Server that it is now primary. The fail-over SQL Server recovers the database and begins accepting client connections. For their part, clients reconnect to the backup server when the primary server fails.

{ewc mvimg, mvimage,!illust.bmp}
Figure 9. Fail-over with Microsoft SQL Server 6.5

The dual-server fail-over is completely automatic. The whole process of detecting and recovering from failure takes only a few minutes. When the primary is repaired, it is restarted and it becomes the new backup server. If desired, the Microsoft SQL Server can fall back to the original server when it is repaired.

Microsoft SQL Server and Windows NT Server provide fail-over by using specialized disk hardware today. This solution will be generalized to use commodity hardware in 1997.

Data replication helps configure and manage partitioned applications. Many applications naturally partition into disjointed sections. For example, hotel, retail, and warehouse systems have strong geographic locality. The applications and databases can be broken into servers for geographic areas. Similarly, customer-care and telemarketing applications often have this strong partitioning. Nonetheless, all these applications need some shared global data. They also need periodic reporting and disaster recovery via electronic vaulting.

Data replication helps solve these problems by automatically propagating changes from one database to another. The replication can propagate changes to a Microsoft SQL Server system at a remote site for disaster recovery. Then, in case the primary site fails, the backup site can recover and offer service.

The same mechanism can be used to allow one site to act as a data warehouse for data capture OLTP systems. The data warehouse, in turn, may publish its data to many data marts that provide decision support data to analysts. Some applications dispense with the data warehouse and have the operational systems publish updates directly to the data marts.

The Microsoft SQL Server data replication system is both powerful and simple to use. A graphical user interface in SQL Enterprise Manager allows the administrator to tell operational databases to *publish* their updates and allows other nodes to subscribe to these updates. This publish-distribute-subscribe metaphor allows one-to-one and one-to-many publication. By cascading distributors, the replication mechanism can be scaled to huge numbers of subscribers. Replication is in transaction units, so each subscriber sees a point-in-time consistent view of the database.

Microsoft SQL Server applications routinely publish tens of megabytes of updates per hour. Publication can be immediate, periodic, or on demand. Replication is fully automatic and very easy to administer.

Microsoft SQL Server has always allowed customers to partition their database and applications among SQL servers running on several nodes. Clients connect to an application at one of the servers. If a client's request needs to access data at another node, the application can either access the data through Transact SQL or make a remote procedure call to the Microsoft SQL Server at the other node.

For example, each warehouse of a distributed application might store all the local orders, invoices, and inventory. When an item is backordered or when a new shipment arrives from the factory, the local system has to perform transactions that involve both the warehouse and the factory SQL servers. In this case, the application running on an Microsoft SQL Server at the warehouse can access the factory data directly or it can invoke a stored procedure at the factory Microsoft SQL Server. OLE Transactions and Microsoft SQL Server will automatically manage the data integrity between the factory and the warehouse.

Once data and applications are partitioned among multiple Microsoft SQL Servers in a cluster, there needs to be a convenient and high-performance way to move data among these servers. Data pipes make it easy to ship data between servers by capturing result sets returned by remote procedure calls directly in node-local tables. This approach can be used by many applications as an alternative to distributed query.

Prior to Windows NT Server 4.0, partitioned Microsoft SQL Server applications had to explicitly manage distributed transactions by calling the Microsoft SQL Server prepare and commit verbs of the two-phase commit protocol. With the new OLE Transactions, applications just declare BEGIN DISTRIBUTED TRANSACTION, and from then on the Distributed Transaction Coordinator (DTC) automatically manages the transaction. DTC is an integral part of Windows NT Server and is one more step toward a full Windows NT Server cluster facility.

The Distributed Transaction Coordinator also connects Microsoft SQL Server to the open transaction standard X/Open XA. Clients can connect to transaction processing monitors such as IBM CICS, Transarc Encina, Novell Tuxedo, and Top End from AT&T Global Information Solutions, which in turn route requests to the Microsoft SQL Servers. The use of transaction processing monitors is another approach to distributing the application by using the TP Monitor to route transactions to the appropriate servers. The TP Monitor also allows Microsoft SQL Server to participate in transactions distributed across many nodes.

Microsoft SQL Server has a companion product, Open Data Services, that is often used as a front-end process to Microsoft SQL Server, either for protocol translation (from foreign protocols to open database connectivity [ODBC] or the database library), or as a simple message dispatcher among multiple SQL servers at multiple nodes.

All these approaches make it relatively easy to partition data and applications among multiple SQL servers in a cluster.

Everything described so far exists. You can acquire and use it today. Indeed, many customers are installing Microsoft SQL Servers, scaling up to four-way SMP hardware systems, and then scaling beyond that by partitioning their databases and applications. Often, these partitions are placed close to the actual data users; data collection is placed in the retail outlets, a data mart of recent activity is placed in the accounting group, and a data warehouse is placed in the planning and marketing group. Each of these applications is fairly separate and partitions naturally. In addition, the data flows among the groups are well defined. Replication and data pipes make it easy to move data among servers. The systems are very scalable. The graphical and operations interfaces combined with Microsoft Visual Basic Scripting Edition are used to automate the operation of many SQL servers.

Over the next few years, Microsoft SQL Server expects to add transparent partitioning of data among SQL servers. This would allow partitioned data without having the application aware of the partitioning — often called *partition transparency*.

After partition transparency is in place, the Microsoft SQL Server group expects to provide *parallel query decomposition*. That is, large data queries typical of decision support applications will be decomposed into components that can be independently executed by multiple nodes of a partitioned database.

It is relatively easy to build huge systems. Microsoft provides easy installation of the operating system and database using graphical tools and wizards. Microsoft SQL Server also includes wizards to set up operational procedures.

But, because these systems involve thousands of client systems and huge databases, *manageability* is the real challenge. Managing and operating a computer system has always been a major part of the cost of ownership. When hardware and software prices plummet, the residual management cost becomes even more significant. Loading, dumping, and reorganizing 100-GB databases is a challenge. At the 3-MB/second data rate of most tape drives, it takes a day and a half just to dump a 100-GB database with one tape drive. Defining and managing the security attributes of 5,000 different users is a daunting task. Configuring the hardware and software for 5,000 clients is another time-consuming task.

Microsoft recognizes that manageability is the largest barrier to scalability. Details of Microsoft's solution to these problems are described in an Enterprise Manager white paper (available in November 1996) that describes Microsoft's Enterprise Manager, and in the product documentation. This section summarizes Windows NT Server and Microsoft SQL Server administrative facilities.

This section includes the following topics:

Managing the software and hardware configurations of thousands of clients is probably the most challenging task of operating large client-server systems. Windows NT Server and Microsoft's System Management Server automate many of these tasks. First, Windows NT Server Security provides a *domain* concept and a single logon for each application running on Windows NT Server. Windows NT Server security provides user groups. Large user populations can be managed by authorizing groups and adding users to groups. The Windows NT Server security mechanism works as a collection of security servers (domain controllers) spread among the network nodes. This distribution provides both scalability and availability. Individual domains have been scaled to over 40,000 users. Windows NT Server security scales beyond that size by partitioning into a multidomain architecture with trust relationships among domains. The security system has both a programmatic and an intuitive graphical interface that allows any node to manage the network security.

Microsoft SMS (System Management Server) allows a single administrator to manage the software configuration, licensing, and upgrades of tens of thousands of clients. SMS automates most tasks and tries to minimize exception situations that only an expert can solve. To give another example, Windows NT DHCP protocol automatically assigns TPC/IP addresses to nodes on demand, thereby eliminating time-consuming and error-prone tasks, allowing node mobility, and conserving the address pool.

Windows NT Server has built-in tools to log errors, manage disk space, set priorities, and monitor system performance. All of these tools can manage a cluster of client and server nodes. Figure 10 shows the performance monitor at one node tracking the CPU and network utilization of several nodes. Each Windows NT Server node has over 500 performance counters for its internal behavior. Microsoft SQL Server, Exchange, and many other products ported to Windows NT Server integrate with the Windows NT performance monitor. Indeed, Microsoft SQL Server adds over 75 performance counters and integrates with the Windows NT Server event log to announce events.

{ewc mvimg, mvimage,!illust.bmp}
Figure 10. Windows NT performance monitor

Microsoft SQL Enterprise Manager is a breakthrough in managing database servers. It gives administrators a visual way to manage and operate many SQL systems from a single console. The graphical interface has several wizards built in to help set up and automate standard tasks. The key features of the SQL Enterprise Manager are:

- A fully graphical administration interface to control and monitor the operation of many Microsoft SQL Servers and the clients accessing them.
- A job scheduler that runs periodic administrative tasks such as dumps, reorganizations, and integrity checks.
- A Distributed Management Objects mechanism that allows administrators to automate exception handling and to automate tasks by writing Visual Basic scripts or letting a wizard write the script. These procedures can use e-mail and a telephony API–based (TAPI-based) beeper system to report results or notify operators when their attention is needed.
- An extension mechanism that allows third parties to add new administrative tools.
- A fully graphical interface to configure and manage database replication.

SQL Enterprise Manager includes wizards to set up routine operations, a wizard to routinely publish Web pages from the database to an Internet or intranet, wizards to help set up a data replication strategy, and answer wizards that guide the operator through the online manuals.

Utilities to load, dump, recover, check, and reorganize large databases are key to operating a system with huge databases. Backing up a 100-GB database using a single high-performance tape drive will take over 33 hours. By using multiple disks and tapes in parallel, Microsoft SQL Server and Windows NT Server have shown a sustained dump rate of 20 GB per hour. This benchmark was I/O limited; with more tapes and disks, the dump rate could have been even faster. The actual management of the dumps can be controlled by the SQL Enterprise Manager job scheduler working with commodity tape robots. These dumps either can be done at full speed or they can be done in the background at a slower rate. By doing incremental dumps and by increasing the degree of parallelism, huge databases can be dumped in a few hours.

Looking at specific examples helps illustrate how Microsoft SQL Server is used in practice.

® **AIM**

AIM Management Group is a large mutual fund manager with over 1,000 employees managing more than $40 billion of assets. AIM uses SQL for both online transaction processing and for data warehousing. The data warehouse stores shareholder account data and mutual fund transaction data. It is growing from 20 GB to 50 GB. The data warehouse receives up to 500 MB of new data each day within a two-hour window. The application runs on a four-way Intel Pentium SMP.

® **Aristotle**

Aristotle Publishing provides a 125 million-record Microsoft SQL Server database containing public voter registration data for the United States. This 90-GB data warehouse application runs on Microsoft SQL Server 6.0 and Windows NT Server 3.51. The server is a dual Intel Pentium 90-MHz server. Over a thousand clients access this database to find voter patterns and preferences. Aristotle is now using Microsoft SQL Server to develop a second application involving 60 million state automobile driver's license records.

® **CITS**

Microsoft CITS is a mission-critical OLTP application key to our product and customer support. It is based on Microsoft SQL Server 6.5 and Windows NT Server 3.51. CITS supports more than 3,600 concurrent users connected via the Microsoft network from all over North America. It has been lab-tested with over 5,000 concurrent users. The database size is about 10 GB, containing customer and product data. It is supported by a 4x100-MHz Pentium Compaq server with 1 GB of RAM and a 52-GB RAID5 disk array. The system is replicated on a hot standby machine in a second building by sending the database log to the standby machine. The standby can take over within 5 minutes. The system provides better than 99.5 percent uptime.

® **The Potomac Group**

The Potomac Group provides Medicaid eligibility verification services to doctors and hospitals in 20 states. The system consists of two 150 MHz DEC Alpha 2100s running Windows NT Server 3.51 and Microsoft SQL Server 6.0. A StorageWorks RAID array stores the 10-GB database. The application software, in combination with DECmessageQ, allows one of the servers to fail over to the other server in a matter of seconds, routing all new transactions to the surviving server. The system has been benchmarked at over 4,000 transactions per minute.

® **TPC-C**

The large TPC-C benchmarks have about 3,500 clients accessing a database of about 80 disks that hold over 300 GB. This includes log space, index information, and metadata. The largest table in a 4,000 tpmC benchmark is the 160-GB OrderLine table with about 3.4 billion 54-byte records. The other tables increase the core database size to about 200 GB. Indices, fragmentation, slack for database growth, and the recovery log increases this to over 300 GB.

Click here to connect to the TPC Web site for more details:

{ewc mvimg, mvimage,!intjump.bmp}

Figure 11 summarizes the current status of TPC-C benchmarks on Windows NT Server platforms.

{ewc mvimg, mvimage,!illust.bmp}
Figure 11. TPC Results On Windows NT Server

Windows NT Server and Microsoft SQL Server scale up to huge databases on a single SMP and scale out to multiple servers, each executing a part of the application and storing a partition of the database. SQL Enterprise Manager makes it easy to configure and manage these servers. OLE Transactions, replication, and data pipes make it easy to move data and requests among them.

Today, a single Microsoft SQL Server can support up to 5,000 active users connected to a single server via ODBC. These servers can process several million transactions in an eight-hour day. They support databases with billions of records spread across one hundred disks, holding hundreds of gigabytes.

Microsoft SQL Server performs these tasks with unrivaled price, performance, and ease of use. The performance of Microsoft SQL Server, Windows NT Server, and the underlying hardware have more than doubled each year for the last three years. We expect this trend to continue for several more years.

The recent additions of OLE Transactions and automatic fail-over are the most recent steps in progress toward Windows NT Server Clusters of Microsoft SQL Servers that can scale both vertically and horizontally.

For the latest information on Microsoft SQL Server, check out the Microsoft SQL Server Forum on the Microsoft Network (GO WORD: MSSQL), or click here to connect to the Microsoft SQL Server Web page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The complete title of this white paper is *Unified Browsing with ActiveX Extensions Brings the Internet to Your Desktop.*

By Stephen Rauch

I don't know about you, but my head is spinning. I've never seen this much technology introduced in such a short period of time. It's all due to the Internet. New development languages are popping up, Internet browsers have introduced a new navigation model, and Microsoft introduced the ActiveX technology in March 1996.

The vision behind Microsoft's ActiveX is unified browsing — accessing information from the Internet just as you do from your local hard drive. Unified browsing means that you don't have to bring up the Windows Explorer to view your file system and another application like Word to view and edit your documents. You get to use one shell to view and navigate through this jungle of information. In addition, it shouldn't matter where that information resides.

In this article, I will introduce you to some of the ActiveX extensions to the Win32 API. These extensions will enable you to write applications, documents, and objects that fit into the ActiveX unified browser context. I'll focus on hyperlinking, which is the navigation method used by Internet browser applications, and a really simple way of downloading data from the Internet with only a Uniform Resource Locator (URL). You don't have to be an OLE expert! Microsoft has hidden all of that OLE stuff under the ActiveX APIs.

This white paper includes the following topics:

Stephen Rauch is a development technical specialist at Reuters. He can be reached on Compuserve at 70313,1455.

The complete title of this white paper is *Leveraging Your Visual C++ Experience on the Internet with Thin Client Technology.*

By David C. Mitchell

There you are, minding your own business, trying to get some real work done, while the rest of the world surfs the Net. You've got nothing against HTML or Web pages, and both Java and ActiveX look promising. But you've been so busy building giant programs using Visual C++, you just haven't had time to sort through all the hype and figure out exactly which tools you're going to use to take these real applications to the Internet.

Let's discuss what you as a developer can do to exploit your Visual C++ experience in the Internet environment. I'll present a way to use the Internet deployment standards offered by Java and still maintain C++ development (including the C++ code that you are currently working on). I'll also focus on the thin client model and why it is best suited for Internet applications. You'll see why traditional client/server models have limited flexibility in the context of the Internet. I'll develop a "litmus test" for determining how effective an application's architecture is at exploiting the Internet. Based on the litmus test, I'll define a framework that allows you to create thin client applications and deploy them over the Internet using Java's cross-platform toolkit, AWT. I'll include various code examples that demonstrate a framework that has already been developed.

A pared-down version of this framework along with support tools to use the framework is available on http://www.msj.com or any of the other sources listed on page 5. The framework and tools supplied are sufficient to develop all of the samples in this article. For this article, I'll focus on the Java solution — in a future article, I'll show you how to use ActiveX to develop a thin client.

This white paper includes the following topics:

David C. Mitchell is the founder and chief technical officer of ViewSoft, a provider of thin client Internet development technology. He can be reached at davem@viewsoft.com or www.viewsoft.com.

Everyone wants to be on the Internet nowadays. This includes grizzled veterans like me who remember when it was the ARPANet. I'm not saying this just to sound more knowledgeable about the subject. It's just that, well, I guess I am just trying to sound more knowledgeable. Anyway, this tidal wave of interest has done a lot for programming. First, it's created a viable market for client programs such as mail readers and Web browsers. Second, it's made programmers like me, who have avoided network protocols and programming with all the ferocity of an Ultimate Fighting match participant, realize that we'd better figure out just what TCP/IP is. (In case you've had your head in the sand for the past year, TCP/IP stands for Transmission Control Protocol/Internet Protocol.)

TCP/IP is simply a suite of protocols. The TCP part describes how two machines set up a reliable connection to each other and transfer data chunks. IP primarily deals with how to get a message routed across the Internet. Each connected machine site has a unique IP address that allows others to figure out a path to any machine around the world. Each machine with an IP address supports numbered ports, each dedicated to a particular service. For instance, you use TCP to send mail via Simple Mail Transfer Protocol (SMTP) using port 21. You retrieve mail with Post Office Protocol (POP) on port 110. These port numbers have been chosen by convention. So a machine needs to run both TCP and IP together (a protocol stack) to communicate with the outside world.

Since the Internet is built on TCP/IP connections, Internet-enabled client programs have to use TCP/IP to connect with remote machines and transfer data in various formats. Fortunately there's a standard API used to write to TCP/IP stacks — sockets. Sockets was originally developed for Unix; Windows Sockets, or WinSock for short, was later developed for Windows-based programs (for more information, see "Plug into Serious Network Programming with the Windows Sockets API" by J. Allard, Keith Moore, and David Treadwell, *MSJ*, July 1993). Here, I'll develop a small OCX that provides selected WinSock functions, then write a 32-bit Visual Basic program that issues a "finger" command, querying a remote machine for information about a specific user.

First, a bit of background on WinSock. When a communications program is WinSock compliant, it usually means that after it dials up and connects with a remote service provider, it provides sockets for your client programs to plug into. Each communications program that implements these TCP/IP services publishes its own implementation of WINSOCK.DLL. These DLLs are almost always incompatible with one another at the server level, although they provide the same external services to clients wishing to connect with them. Therefore, if you overwrite another communication program's WINSOCK.DLL, that program won't work correctly, but a client program such as my finger program (discussed later) will work the same way with any communication package's WINSOCK.DLL. If that seems confusing to you, you've had more luck in avoiding these sorts of problems than many others have had.

The other thing to know is that a program can provide a 16-bit or a 32-bit WinSock implementation. If your communications program is 16-bit, you can't run a 32-bit WinSock client with it. However, if you have a 32-bit comm program or dialer (such as the Windows 95 Dial-Up Networking), you can use both 16- and 32-bit clients with it. While developing this month's programs, I used my account on Netcom, a popular service provider. Netcom provides only a 16-bit dialer, NetCruiser, but by setting up the Windows 95 dialer instead, I was easily able to start connecting through a more reliable 32-bit connection.

One of the biggest concerns in developing a WinSock program is that not all data requests are fulfilled immediately. If you're awaiting a return or an acknowledgment from a distant machine, it can take several seconds or even minutes before the messages make their way back across the Internet. For this reason, WinSock functions are divided into two groups, synchronous (returning data immediately) or asynchronous (you might have to wait for a return message). Synchronous messages include those that don't need to go across the network for return data, such as network-to-client byte-order conversion calls. Asynchronous calls include calls like remote connections, where you won't know whether it worked right away.

If your program sends a message to a machine halfway around the world and sits there waiting for a reply, it might appear to be hung. You should write your own handler to process other Windows messages during the wait time, or you'll risk freezing everyone's program while your selfish little app waits for incoming data (in 16-bit Windows, anyway), a problem often known as the WinCIM Syndrome.

When you have to make a WinSock call that requires asynchronous operations, the call itself handles it for you. Last month in my Visual Programmer column, I discussed that many callback functions in Windows 95 are designed to expect a window handle and message ID. Asynchronous WinSock calls use this mechanism. If you're expecting to receive some data from halfway around the world, you can call the WSAAsyncSelect function, and pass it a window handle, message ID, and a flag indicating you want to wait for data:

```
WSAAsyncSelect(socketID, hWnd, WM_USERDEFINED, FD_READ)
```

When the data actually arrives from somewhere on the network and hits your TCP/IP stack, your window's message handling procedure is notified. Meanwhile, your interface doesn't freeze and you can go do other processing.

Now suppose you want to do this all in Visual Basic. Since you can't use Visual Basic to capture user-defined messages within a Visual Basic form's code, you need to give it a bit of help. You need a WinSock OLE control that can handle these asynchronous calls and trigger events in Visual Basic when something happens.

This white paper includes the following topics:

To enable the Windows NT Guest account

1. Run Windows NT and start the User Manager utility.
2. Select **Guest** in the list of accounts.
3. Click **Properties** on the **User** menu.
4. Clear the **Account Disabled** check box and click OK.

The Internet represents the dawn of a new era of global communications and wide-area public networks. It is a technological revolution at least as important as the development of the personal computer. Microsoft believes that businesspeople, consumers, students, and users of all types can benefit from a full integration of these two revolutions — the Internet and the PC. Microsoft's strategy is to work openly with customers, other companies, and industry groups to realize the full potential of the Active Internet. The next generation of powerful Internet applications promises to make the Internet more exciting and useful by seamlessly integrating audio, video, 3-D animation, and more.

Microsoft's strategy to support business on the Active Internet is exactly the same as its overall distributed-computing strategy — to provide the network foundation for distributed computing (Microsoft Windows NT Server); a family of integrated server applications (BackOffice); distributed object and systems technology (Win32 and OLE); a complete set of development and authoring tools (Visual Basic, Java, Java Script, Front Page, Internet Studio); and a set of desktop applications (Office) and systems (Windows 95 and Windows NT Workstation) that take advantage of all these features for an integrated view of the world.

Many organizations have already realized the benefits of the Internet as a means to rapidly publish information. Companies today are using the World Wide Web primarily as an electronic storefront, billboard, or Yellow Pages ad. Many have applied these same principles to *internal* communication, establishing *intra*nets, or private Web-based networks to exchange mission-critical information across local, regional, and even global enterprises. Until now, this information has been predominantly static, preformatted Hypertext Markup Language (HTML) documents, not interactive, user-centric applications.

The same forces that have driven the evolution of client/server technology are now at work driving the Internet toward a more dynamic and interactive environment. Whether in traditional distributed computing or on the Internet, businesses and users need to access and manage an ever-expanding amount of information, from anywhere, at any time.

By connecting Internet or intranet Web servers to relational databases like Microsoft SQL Server, many organizations have moved beyond static document publication to an Active Internet, where content is generated dynamically from a database, automatically or in response to a user request. An Active Internet application may use electronic forms to collect orders, customer information, and other transactions, and then store them in a relational database. The database, in turn, allows processing, retrieval, and analysis with a variety of off-the shelf-tools. An Active Internet can store information in a single, secure place — without duplication — regardless of whether that information is accessed internally or over the World Wide Web.

Progressive companies are driving the Web in a new and even more interesting direction, where business can be conducted over the Internet. These companies are building a new class of Active Internet business applications, based on products like Microsoft Internet Information Server, Microsoft SQL Server, and the BackOffice family of products.

The Active Internet is a new type of distributed client/server environment that extends beyond traditional corporate IT boundaries. A major requirement for an Active Internet is a secure, reliable, and scaleable place to store information. Microsoft SQL Server was built from the ground up as a scaleable, high-performance database management system, designed specifically for the unique requirements of distributed client-server computing. SQL Server has been fulfilling this role in business-critical environments for several years.

As the Internet and intranets become more complex and more heavily trafficked, managing incoming and outgoing information becomes even more difficult. By applying basic distributed computing and client/server principles to this new frontier, companies can reduce the risk and the complexity of doing business on the Internet. Integrating Microsoft SQL Server with the Internet or an intranet allows organizations to:

- **Build Active Web sites,** capturing the user's interest by publishing real-time information and providing user interaction and customization.
- **Conduct business on the Internet,** securely and reliably.
- **Develop corporate intranets,** giving users new tools to access business information, without compromising security and data integrity.

This section includes the following topics:

From its foundation of superior performance, reliability, and scalability to its tight integration with Microsoft Internet Information Server and the other BackOffice products, Microsoft SQL Server is unquestionably the best database platform for the new breed of Internet applications. Microsoft SQL Server version 6.5 builds on this success by introducing a number of tools and technologies that provide significant advantages over alternative solutions:

® A fast and easy programming interface

® Open Internet database connectivity

® Automated Web publishing of database information

® Cost-effective licensing

A Fast and Easy Programming Interface

Traditionally, Common Gateway Interface (CGI) scripts have been used to add functionality to standard HTML forms, such as linking forms with a SQL database. Each field on an HTML form may require 25 lines of CGI script to link it with a database. This model requires a high level of up-front programming. Any modifications are extremely labor intensive, given the one-to-one correspondence between the form field and the CGI script. The lack of quality database connectivity tools is partially responsible for the slow migration of commercial and business applications to the Internet.

Figure 1 graphically demonstrates how the Internet Database Connector (IDC) and the SQL Server Web Assistant enable seamless connectivity with SQL Server data. The IDC, or *pull model,* allows a Web user to initiate dynamic user driven queries from within an HTML document to a SQL Server, which retrieves or updates information in a database. The SQL Web Assistant, or *push model,* allows a database administrator or Webmaster to define a set of data (a query or stored procedure), which is automatically merged into an HTML document either on a scheduled basis or when the actual data changes.

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. Microsoft provides two powerful and complimentary tools for integrating SQL Server with the Web — the Internet Database Connector (IDC)and SQL Server Web Assistant

Open Internet Database Connectivity

The IDC is an Internet server application programming interface (ISAPI) application that uses Open Database Connectivity (ODBC) application programming interfaces (APIs) to send and retrieve information between SQL Server and the Internet. The IDC provides the ability to create direct links between fields on HTML forms and SQL Server data without the need for complicated CGI scripts. Organizations are already using the Internet Database Connector to:

® Publish real-time database information to the Active Internet

® Capture and store gigabytes of information directly into Microsoft SQL Server

® Perform high-performance searches of online Microsoft SQL Server data

® Build a new class of Active Internet commercial and business applications

Conceptually, database access is performed by Internet Information Server as shown in Figure 2.

{ewc mvimg, mvimage,!illust.bmp}
Figure 2. The Microsoft Internet Information Server and the IDC allow Internet users access to databases.

The Internet Database Connector uses two types of files to control how the database is accessed and how the output Web page is constructed. These files are Internet Database Connector (.idc) files and HTML extension (.htx) files.

The Internet Database Connector files contain the necessary information to connect to the desired ODBC data source and execute the SQL statement. An Internet Database Connector file also contains the name and location of the HTML extension file.

The HTML extension file is the template for the actual HTML document that will be returned to the Web browser after the database information and the extension file have been merged. This allows static text, graphic images, or even real-time video to be combined with database information on the same Web page.

The Internet Database Connector is ideal for building a new class of Active Internet applications that allow the user to view and update information in a database. Applications like event registration, loan application processing, and on-line banking can be created using this technology.

Automated Web Publishing of Database Information

Microsoft SQL Server Web Assistant, one of the new Internet extensions available in Microsoft SQL Server 6.5, provides the ability to easily and automatically generate Web pages formatted in HTML. Used in conjunction with an appropriate Web server product, the Web Assistant publishes Microsoft SQL Server-based information from anywhere in an organization.

The Web Assistant uses a step by step, wizard-like interface that walks a DBA or Webmaster through the process of creating a query, formatting output, and scheduling the query execution — making it easier than ever for customers to create interactive, database-driven Web sites with automated information content. Using a data "push" model, the Web Assistant automatically publishes information out of a SQL Server database directly to the Web page. The data can be updated automatically — on a regular scheduled basis or whenever relevant data changes. The data remains accessible to any existing client/server application as well as to the Web.

The Web Assistant is particularly valuable for publishing large amounts of database information that will be the same for all visitors to a Web site, such as price lists, inventory or on-line catalogs. The Web Assistant is helpful in publishing time-sensitive or dynamic data, where users need to view it in real-time, such as stock quote information. The Web Assistant is resource-efficient, executing the query only once, regardless of how many people visit the page.

Cost-effective Licensing

Traditional client/server relational database management system (RDBMS) user-based licensing models, where customers are charged for the number of concurrent users accessing the database, have proven unworkable for the Internet. The Microsoft SQL Server Internet Connector provides a simple, flexible, and cost-effective way for customers to license their databases for use on the Internet or corporate intranets.

The Microsoft SQL Server Internet Connector license is an open solution, providing access to Microsoft SQL Server from the Microsoft Internet Information Server or from third-party Web servers. The SQL Server Internet Connector allows an unlimited number of Internet and intranet connections to a single SQL Server.

The Microsoft SQL Server Internet Connector license will support customers using either Microsoft SQL Server version 6.0 or Microsoft SQL Server version 6.5. For more information, see "Microsoft SQL Server Internet Connector Q & A" at http://www.microsoft.com/SQL/netqa.htm.

For more than two years, Microsoft has been engaged in strategic development for the Active Internet, with the goals of making it easy for customers to get on the Internet, making it safe and reliable for significant business and consumer applications, and creating a viable business model that will support the industry and its consumers.

As a result, Microsoft is well positioned to devise and implement a comprehensive Active Internet strategy to meet the needs of corporate users, consumers, content creators, software vendors, Web publishers, and other information providers for a unified computing and Internet experience.

To deliver the full potential of the Active Internet, Microsoft recently introduced ActiveX technologies, a robust framework for creating interactive content using software components, scripts, and existing applications that can be embedded in HTML documents. Specifically, ActiveX technologies enable developers to build Web content easily using ActiveX controls (formerly OLE Controls), ActiveX scripting, and ActiveX documents.

The ActiveX server framework allows developers to create rich, interactive applications for Microsoft BackOffice, leveraging their existing experience, knowledge, and tools. ActiveX server controls and ActiveX server scripts, written with a variety of scripting languages like Visual Basic Scripting Edition, PERL, and JavaScript, become the building blocks for the next generation of server-driven active content. Together, these ActiveX controls and scripts make it even easier for organizations to integrate Microsoft SQL Server, and the other BackOffice products, with the Internet for building active Web sites, conducting business, and developing corporate intranets.

As businesses explore new ways to leverage the Internet for business operations, they have been quick to recognize that most of the business content they would like to publish is already being managed by a relational database management system. These businesses also realize that today's Web sites are measured not on content alone, but also on presentation and the level of interaction. The more popular sites capture the Web user's interest by publishing real-time information and building customized pages. These sites put users in control, allowing them to browse large volumes of data and provide input. It takes a client/server database like Microsoft SQL Server to accomplish this.

Ethos Corporation, for example, is an interactive service provider delivering up-to-the-minute financial news and research information to more than 300,000 online investors weekly. Their Web site, *Investors Edge* at http://www.investorsedge.com/, is among the most active financial services on the Internet, offering online stock quotes, corporate summaries, financial news, and personal portfolio analysis. Microsoft SQL Server drives this site, collecting, managing and publishing the real-time financial data while averaging 3.4 million transactions per day.

Personal portfolio analysis brings a new level of customization to the *Investors Edge* Web site, allowing each user to build an individual portfolio of 15 stocks to be tracked over time. Once established, a portfolio is only accessed with a unique user id and password combination, displaying a completely personalized Web page of the user's stock performance history. This sensitive information is protected by the integrated C-2 security of Windows NT Server and SQL Server.

Many organizations are already realizing the benefits of *extending the enterprise,* that is, connecting existing business systems to the Internet. By automating the communication and processes between customers and suppliers, businesses become more efficient. But electronic commerce is progressing more slowly than expected. Fewer than 10 percent of Internet users make regular online purchases. The biggest deterrents for conducting business on the Internet are data integrity and security.

Security on the Internet is critical to both extending the enterprise and conducting electronic commerce. The industry has focused on defining technologies and industry standards to secure the communications between Web servers and browsers. Technologies and standards include encryption, secure transaction, user authentication, and signed-code or digital signatures. Microsoft and its partners have participated in the development of several important new technology standards, including Secure Sockets Layer (SSL), which is a secure protocol providing data encryption, server authentication, and message integrity for a TCP/IP connection, and Secure Electronic Transactions (SET), which is a technical standard for safeguarding payment card purchases made over open networks like the Internet.

While these are critical components in providing a secure environment for conducting business on the Internet, they only address part of the security problem. Looking at traditional client/server environments gives insight into other security issues. Controlling user or application access to information is a fundamental security requirement of any business application. Data and transaction integrity are also critical in any multi-user business environment, ensuring that all transactions are either fully completed or are rolled back so that every user's view of data is consistent and reliable.

These security and integrity issues were solved for traditional business applications years ago by Microsoft SQL Server technologies, such as user and application security, data integrity and concurrency controls, datastream encryption, data replication, and transparent distributed transactions. Solving these issues for the Internet doesn't require new technology. Applying SQL Server technology to the Internet gives you proven solutions to the challenges of data security and integrity.

Data Track Systems, Inc.(http://www.datatrac.com/), a Value Added Network provider specializing in the electronic ordering and receipt real estate settlement services, is using Microsoft SQL Server to solve the challenges of conducting business transactions on the Internet. Data Track Systems is deploying an Internet application built on Microsoft SQL Server, easily allowing real estate agents to order and track title, credit, property valuation, escrow, and home inspection services. By providing open and secure Internet access between requesters and vendors along with integrated billing services, Data Track Systems creates efficiencies in ordering, processing, and delivery of settlement documents, resulting in cost reductions and faster turnaround.

Controlling User and Application Security

Internet and intranet servers both need to augment anonymous user control with specific per-user and per-group access control. Microsoft Internet Information Server and Microsoft SQL Server both build on the Windows NT Server security model to deliver Internet-specific security features, such as control over anonymous access, network or host-based access, and secure Internet authentication.

Consider a group of intranet users who have been given access to a particular Web site. With client authentication, users can identify themselves to Internet Information Server and gain access to those directories and files marked as readable only by the group. Because of the full-featured Windows NT Server user accounts database, any number of overlapping groups can be created and granted differing access to resources on the server. Taking this a step further, if the groups are authenticated by Windows NT Server, they are also authenticated to SQL Server; therefore you don't have to maintain multiple security databases.

Replicating Rich Text and Graphical Content

Microsoft SQL Server replication solves the traditional client/server challenge of distributing and synchronizing information throughout an organization, while guaranteeing integrity and reliability. Replication solves a similar set of problems for Internet and intranet sites, allowing Webmasters to manage and synchronize multiple copies of published information. Replication also allows an organization to avoid exposing sensitive database information on an intranet by selectively replicating data from production SQL Servers to a Web-connected SQL Server. Microsoft SQL Server version 6.5 can replicate complex text and image data popular on Web sites.

Integrating Legacy Systems

Most medium-to-large companies today have significant investments in their information systems infrastructure.

Information is stored in a variety of different, and often incompatible, database management systems. The vast majority of these legacy systems provide no integration with the Internet. With Microsoft SQL Server compatible middleware solutions like Microsoft TransAccess or Information Builders Inc.'s EDA/Open Database Gateway, customers can integrate data across the enterprise onto the Internet.

Business use of internal Internets, called i*ntranets*, will grow as corporations find them the best foundation for wide area networks. Intranets let colleagues work together, whether they're around the corner or around the globe. While exact figures are hard to come by, industry analysts estimate *intra*net web sites outnumber Internet Web sites ten-to-one — an amazing growth rate considering *no* intranets existed three years ago!

To support this shift, Microsoft is making internal business use of the Active Internet a key strategic focus. For example, on the desktop, Microsoft is making Windows clients (Windows 95 and Windows NT Workstation) the best Active Internet clients by designing Internet capabilities into the operating systems themselves. Microsoft has enhanced Office — the leading choice in business productivity software — so that Office tools serve as Web creation tools. Microsoft has also offered free viewers, which let users read Office documents, even if they don't have Office installed on their local PCs. Microsoft has integrated Web protocols — TCP/IP and HTTP — into its server platform (Windows NT Server), allowing Internet users to share, search, and use Web pages and documents the same way they use files and applications on the LAN. With Microsoft SQL Server 6.5, Microsoft is extending its industry leading distributed client-server database to be the best engine for corporate intranet applications.

Business applications on corporate intranets have the same requirements for security and integrity as their Internet counterparts. Organizations developing and deploying intranet applications are benefiting from the same proven technologies — user and application security, data integrity and concurrency controls, datastream encryption, data replication, and transparent distributed transactions — that make Microsoft SQL Server the best platform for conducting business on the Internet.

George Weston, for example, went live in February with an intranet employee benefits application supporting over 1,500 users. With SQL Server as the content engine, Weston has a scaleable and secure system that is easy to support and maintain. Also, since the information is being published from SQL Server, Weston employees responsible for keeping information up-to-date don't have to become an expert in HTML. They can continue to use familiar tools like Microsoft Access to make necessary changes.

Before jumping into these new APIs, I want to briefly introduce the foundation that these APIs are based on, URLs and URL monikers. For more information, you may wish to read the asynchronous URL moniker specifications distributed with the ActiveX SDK.

A URL is a string representation of the name and address of an object on the Internet. There are two parts to the syntax of a URL. The first part is the name of a protocol, and the second part is protocol-specific. For a Hypertext Transfer Protocol (HTTP) URL, the protocol is HTTP, which is followed by a colon. The protocol-specific part after that is two slashes, then a host name, an address, and an optional port number. The whole thing comes out like this: http://www.microsoft.com. The nice thing about the URL syntax is that it is extensible. Just about any protocol you define can be encoded as a URL. Examples include File Transfer Protocol (FTP) and Gopher. In addition, the URL syntax is easily read and understood.

To actually do something with a URL, you need to write some protocol-specific code. This is where URL monikers come into the picture. A moniker is just a symbolic name of an object; encapsulated in that name is the mechanism to instantiate the object. The moniker architecture is extensible and supports complete name parsing through the MkParseDisplayName(Ex) API and the IParseDisplayName and IMoniker interfaces. It also supports human-readable names through the IMoniker::GetDisplayName method. When you use monikers, you are actually leveraging an extensible infrastructure that takes care of getting tasks done, such as downloading files, finding and launching code, or encoding/decoding raw data into appropriate formats.

URL monikers provide the framework for building and using URLs. Unlike the synchronous monikers defined in the OLE 2.0 specification, ActiveX URL monikers support asynchronous binding as well. They are implemented asynchronously because URLs frequently refer to resources across low bandwidth, high-latency networks. You don't want your applications or controls to block.

Figure 1 shows the three components involved in using URL monikers: client, system, and transport. You implement the client component in your applications. The system component is provided by the operating system. The third component is transport-specific.

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. URL Moniker Components

To use URL monikers, you must implement the IBindStatusCallback interface, a callback object used by the URL moniker to provide feedback to the client. It receives progress notification through the IBindStatusCallback::OnProgress member function, data availability notification through IBindStatusCallback::OnDataAvailable, and various other notifications about the status of the binding from the moniker. You can use this information to provide feedback to your users through a progress bar with a cancel button or through progressive rendering of data.

You may optionally implement a format enumerator with the IEnumFormatETC interface. This interface lets you provide protocol-specific information that will affect the bind operation. For example, to support media-type negotiation, you can implement IEnumFormatETC to register a MIME format enumerator. A URL moniker translates these formats into MIME types when binding to HTTP URLs.

Let's take a look at what happens when you use a URL moniker. The first thing you do is create a bind context by calling the ActiveX API CreateAsyncBindCtx. You pass this function a pointer to the IBindStatusCallback interface you implemented, and optionally a pointer to the IEnumFormatETC interface. CreateAsyncBindCtx returns a pointer to IBindCtx, which automatically registers the IBindStatusCallback and the IEnumFormatETC interfaces with the bind context. Once you have a bind context, you can create a URL moniker by calling MkParseDisplayNameEx or CreateURLMoniker. OLE's MkParseDisplayName function, which converts a string into a moniker that identifies the object named by the string, has been extended to support URLs. These two steps are essentially the same steps you use today to create a file or item moniker.

After you create the bind context and the URL moniker, you call BindToStorage or BindToObject on the URL moniker, passing the bind context. When either of these functions are called, the URL moniker figures out if this is an HTTP binding or an FTP binding and instantiates a transport-specific component. How does the URL moniker know which transport protocol to use? It looks at the protocol part of the URL syntax you passed when you created the moniker using MkParseDisplayNameEx or CreateURLMoniker.

Once the transport-specific component is created, the IBinding interface is passed back to you through a callback, IBindStatusCallback::OnStartBinding. With the IBinding interface, you can stop, suspend, and resume the binding operation or change the priority of the download. From this point on, the transport-specific component is driving the bind operation, providing progress and data callback notifications to the client. The bind operation ends when you get your last IBindStatusCallback::OnDataAvailable callback (a parameter lets you

know) or when you get the IBindStatusCallback::OnObjectAvailable callback if you are binding to an object.

Even through this was a brief description of how to create and use URL monikers, it sounds like a lot of work. If you want something simpler, you may want to use the APIs Microsoft created for hyperlinking and downloading data based on this infrastructure.

You see an ad for a cool-looking upcoming movie and decide to check out the Web site with info on it. You enter a string in your favorite World Wide Web browser and presto, you're viewing an HTML document. You continue to navigate your way through the Internet, clicking on some underlined text displayed in an HTML document or some cool graphics, and before you know it, you've eaten up six hours. You have to go to work in three hours and your connect-time charges are pricier than a Pentagon toilet seat.

Hyperlinks permit you to explore vast amounts of information without caring about the location of the server or the name of the file you want to view. All you need to do is enter a URL or point-and-click on some underlined text displayed in an HTML document and your browser is off to a new location displaying something new. World Wide Web browser applications have other navigational features as well. Let's face it, as you browse the Internet, you can easily forget where you came from. That's why browsers provide a history of places you've been, Go Forward and Go Back buttons, and a favorites list.

Microsoft extended the hyperlinking metaphor by introducing ActiveX hyperlinks. Like other OLE-based integration technologies, ActiveX hyperlinks allow you to integrate them seamlessly with other applications that support hyperlinks. A few different forms of navigation include:

® From a standalone application to another standalone application, in the absence of a browser.

® From a standalone application to an ActiveX document object or HTML document in a browser.

® From one ActiveX document object in a browser to another ActiveX document object in the same browser, analogous to navigating from one HTML document to another.

® From an ActiveX document object in a browser to an ActiveX document object in a Microsoft Office Binder-like application if the hyperlink target is embedded in an Office Binder document.

® From one location in an object/document to another location in the same object/document.

The ActiveX hyperlink functions and interfaces support both ActiveX document and non-ActiveX document applications. You can also create hyperlinks to jump from standalone, non-OLE applications to HTML documents and back again. Depending on the complexity of the hyperlinking you need to include in your documents and applications, you have two choices: the ActiveX Simple Hyperlink Navigation APIs or the ActiveX Hyperlinking interfaces.

The Simple Hyperlink Navigation APIs (see **Figure 2**) are ActiveX extensions to the Win32 API. They encapsulate the more complex full hyperlinking interfaces, which I will discuss later. If you can say “Here is where I want to go, now go there” without needing to know any OLE interfaces and objects, you want to use these APIs.

Figure 2. Simple Hyperlink Navigation APIs

Function	Description
HlinkSimpleNavigateToString	Executes a hyperlink jump to a new document or object represented as a string
HlinkSimpleNavigateToMoniker	Executes a hyperlink jump to a new document or object represented as a moniker
HlinkGoBack	Executes a hyperlink jump backwards within the navigation stack
HlinkGoForward	Executes a hyperlink jump forwards within the navigation stack

For example, if you’re a control writer and if you want your control to hyperlink to one HTML document while embedded in another HTML document, you could use these APIs to get the job done. Another use would be in an image map object; when users click on areas in the image map, you can link them to sites represented by the areas of the image map. These APIs will be extended in the future to provide a simple method of accessing other hyperlink navigation metaphors such as histories and favorites.

Let’s see these APIs in action. I modified the Bindable Scribble sample application (BINDSCRB) distributed with Microsoft Visual C++ 4.1 so that it displays a context menu. A context menu is nothing more than a popup menu displayed when Windows sends a WM_CONTEXTMENU message to your application. I chose the BINDSCRB application because calling the Simple Hyperlink Navigation APIs from within an application whose documents are displayed in-frame results in different behavior than if you call the APIs from a standalone application. The BINDSCRB document is an ActiveX document object; its documents can be hosted within any hyperlink frame application (see **Figure 3**), including Microsoft Internet Explorer 3.0 (IE 3.0).

{ewc mvimg, mvimage,!illust.bmp}
Figure 3. BINDSCRB sample application.

The Simple Hyperlink Navigation APIs are defined in the URLHLINK.H header file from the ActiveX SDK beta available at http://www.microsoft.com/intdev/. You must also link with the URLMON.LIB and URLHLINK.LIB static libraries. These are static libraries for now; this functionality will be folded into URLMON.DLL in the next beta release.

To add the context menu support to the BINDSCRB application, I added a popup menu to SCRIBBLE.RC.

```
IDR_CONTEXTMENU MENU DISCARDABLE
BEGIN
    POPUP ""
    BEGIN
        MENUITEM "Go &Back",      ID_GOBACK
        MENUITEM "Go &Forward",   ID_GOFORWARD
        MENUITEM SEPARATOR
        MENUITEM "&Microsoft Home page", ID_MSHOMEPAGE
    END
END
```

Selecting of the Go Back and Go Forward menu items causes BINDSCRB to jump backwards and forwards within the browser’s history list. Selecting “Microsoft Home page” will hyperlink to a Scribble HTML document residing on the Internet (actually http://www.microsoft.com/default.htm).

I created a message handler using ClassWizard for the WM_CONTEXTMENU window message in the CScribView class (see **Figure 4**). This message is received by the BINDSCRB application when the user right-clicks anywhere in its view.

Figure 4. WM_CONTEXTMENU Message Handler

```
void CScribView::OnContextMenu(CWnd* pWnd, CPoint point)
{
    CRect rect;
    GetClientRect (&rect);
    ClientToScreen (&rect);

    if (rect.PtInRect (point))
    {
        CMenu menu;
        menu.LoadMenu (IDR_CONTEXTMENU);
        CMenu* pContextMenu = menu.GetSubMenu (0);

        // If the Scribble application is not hosted in a frame that
        // is capable of hosting ActiveX Document Objects, disable the
        // "Go Back" and "Go Forward" menu items.  The Simple
        // Hyperlink Navigation APIs related to Go Back and Go Forward
        // only work in-frame.
        CScribDoc* pScribDocument = GetDocument ();
        if (!pScribDocument -> IsDocObject())
        {
            pContextMenu -> EnableMenuItem(ID_GOBACK, MF_GRAYED);
            pContextMenu -> EnableMenuItem(ID_GOFORWARD, MF_GRAYED);
        }

        // Display the context menu.
        pContextMenu -> TrackPopupMenu (TPM_LEFTALIGN | TPM_LEFTBUTTON |
                                        TPM_RIGHTBUTTON, point.x,
                                        point.y, this);
    }
}
```

I call the CScribDoc member function IsDocObject to determine if the view is hosted in a container that supports the IOleDocumentSite interface. Containers that support this interface are generally hyperlink-aware and will host ActiveX doc objects. IE 3.0 and Microsoft Office Binder are examples. If the view is not hosted in such a container, I disable the Go Back and Go Forward menu items. HlinkGoBack and HlinkGoForward may be called only if the object calling the APIs is hosted within a hyperlink-aware container.

To handle the individual context menu items, I added three message handlers to SCRIBDOC.CPP (see **Figure 5**). CScribDoc::OnGoBack and CScribDoc::OnGoForward are invoked when the user selects the Go Back and Go Forward menu items, and CScribDoc::OnMSHomePage is invoked when the user selects “Microsoft Home page.”

Figure 5. Context Menu Item Message Handlers

```
////////////////////////////////////////////////////////////////////
//
//    OnGoBack: Context menu handler for "Go Back".  Calls the
//              Simple Hyperlink Navigation API HlinkGoBack to go
//              back in the Hyperlink history stack.
//
////////////////////////////////////////////////////////////////////
void CScribDoc::OnGoBack()
{
  IUnknown* pUnk = NULL;

  // Obtain the IUnknown pointer to the document.
  HRESULT hr = this -> ExternalQueryInterface ((void*)&IID_IUnknown,
```

```
                                                LPVOID *)&pUnk);

  if (hr = = S_OK)
    // Go Back in the hyperlink history stack
    HlinkGoBack (pUnk);
}

////////////////////////////////////////////////////////////////
//    OnGoForward:  Context menu handler for "Go Back".  Calls
//                  the Simple Hyperlink Navigation API HlinkGoForward
//                  to go forward in the Hyperlink history stack.
//
////////////////////////////////////////////////////////////////
void CScribDoc::OnGoForward()
{
  IUnknown* pUnk = NULL;

  // Obtain the IUnknown pointer to the document.
  HRESULT hr = this -> ExternalQueryInterface ((void*)&IID_IUnknown,
                                                LPVOID *)&pUnk);

  if (hr = = S_OK)
    // Go Forward in the hyperlink history stack
    hr = HlinkGoForward (pUnk);
}

////////////////////////////////////////////////////////////////
//
//    OnMSHomePage:  Context menu handler for "Microsoft Home Page".
//                   Calls the Simple Hyperlink Hyperlink Navigation
//                   API HlinkSimpleNavigateToString to display the
//                   HTML page for Microsoft's Web site
//
////////////////////////////////////////////////////////////////
void CScribDoc::OnMSHomePage()
{
  HRESULT hr;
  IUnknown* pUnk = NULL;

  if (this -> IsDocObject())
    // Obtain the IUnknown pointer to the document.
    hr = this -> ExternalQueryInterface ((void*)&IID_IUnknown,
                                         LPVOID *)&pUnk);

  HlinkSimpleNavigateToString(L"http://www.microsoft.com/default.htm",
                              NULL, NULL, pUnk, 0, NULL, NULL, 0);
}
```

When calling HlinkGoBack and HlinkGoForward, you must pass the IUnknown pointer of the document or object initiating the hyperlink. For an aggregated COM object, this must be punkOuter. This information is used to traverse the ActiveX hyperlinking interfaces to determine where the object resides in the history list. Once its position has been established, the hyperlinking architecture navigates from that position in the list backward or forward as appropriate.

You can declare where you want to go using the HlinkSimpleNavigateToString or HlinkSimpleNavigateToMoniker APIs. If you have a string that identifies the hyperlink target, such as http://www.microsoft.com, you will use HlinkSimpleNavigateToString. Pass the string as the first argument; it will be automatically resolved into a moniker by the API for underlying binding operations by calling MkParseDisplayNameEx internally. If you have a

moniker that identifies the hyperlink target, you will use HlinkSimpleNavigateToMoniker. The moniker passed can be of several different types: a URL moniker, a file moniker, or an item moniker. For both of these API functions you can also pass NULL as the first argument, in which case the navigation is considered within the same document or object.

The remaining arguments to these two APIs are the same.

```
STDAPI HlinkSimpleNavigateToString
           (LPCWSTR szTarget,LPCWSTR szLocation,
            LPCWSTR szTargetFrameName,
            IUnknown *pUnk,IBindCtx *pbc,
            IBindStatusCallback *pbsc,
            DWORD grfHLNF, DWORD dwReserved);

STDAPI HlinkSimpleNavigateToMoniker
           (IMoniker *pmkTarget, LPCWSTR szLocation,
            LPCWSTR szTargetFrameName, IUnknown *pUnk,
            IBindCtx *pbc, IBindStatusCallback *pbsc,
            DWORD grfHLNF, DWORD dwReserved);
```

szLocation is an optional string representing the location within the hyperlink target for the new hyperlink. For an example of this, bring up an HTML document that uses hyperlinks for a table of contents. When you select a table-of-contents hyperlink, you jump to a different location within the same HTML document. szTargetFrameName is an optional string that describes the target frame for the hyperlink when navigating within a document container that supports the IHlinkFrame interface. The pUnk argument is a pointer to IUnknown, as described earlier. It is used by the hyperlink architecture as a reference to the document or object in the history list. If pUnk is NULL, it is assumed that the hyperlink originates from an OLE-unaware application, in which case the APIs will call ShellExecute to open the szTarget or pmkTarget document or object in a newly created instance of the browser.

In my CScribDoc::OnMSHomePage implementation (see **Figure 5**), I call the CScribDoc class's member function IsDocObject to determine if the view is hosted in a container that supports the IOleDocumentSite interface. If it is not, I leave pUnk set to its initialized state, NULL. When the user is running the BINDSCRB application standalone and selects the Microsoft Home page menu item, the hyperlinking architecture creates a new instance of IE 3.0 (see **Figure 6**). If the BINDSCRB document is hosted in a hyperlink-aware container, I set pUnk equal to the IUnknown pointer of CScribDoc, whose base class is CDocObjectServerDoc. When the user selects the Microsoft Home page menu item, a new instance of IE 3.0 is not created. Instead, the browser containing the BINDSCRB document hyperlinks and displays the requested target, in this case http://www.microsoft.com/default.htm.

{ewc mvimg, mvimage,!illust.bmp}
Figure 6. Hyperlinking from BINDSCRB

The grfHLNF argument is a value from the HLNF enumeration defined in the header file HLINK.H. Values from the HLNF enumeration indicate how hyperlink navigation is to proceed. For example, if you set grfHLNF equal to HLNF_OPEN_INNEWWINDOW, the hyperlinking architecture will create an instance of IE 3.0 to display the navigation target. You can see HLNF_OPEN_INNEWWINDOW in practice today if you right-click on a hyperlink displayed in an HTML document by IE 3.0 and select Open In New Window. The values in the HLNF enumeration also convey context information about the navigation from each of the objects participating in the navigation protocol to the other objects. The pbc argument is a pointer to a bind context to use for any moniker binding performed during the navigation. The bind context is an object that stores information about a particular binding operation. You can pass the pointer to the function if you wanted to add additional arguments on the IBindCtx such as a format enumerator. Finally, the pbsc argument is a pointer to an IBindStatusCallback interface. You would pass a pointer to this interface if you are interested in progress notification, cancellation, pausing, or any low-level binding information.

To make things even simpler, because many of the arguments that you pass to HlinkSimpleNavigateToString or HlinkSimpleNavigateToMoniker will probably be NULL, Microsoft also provides two macro equivalents, HlinkNavigateString and HlinkNavigateMoniker. HlinkNavigateString has two arguments, a pointer to IUnknown and the string identifying the hyperlink target. HlinkNavigateMoniker also has two arguments, a pointer to IUnknown and a pointer to a moniker that identifies the hyperlink target.

Based on what I have covered so far, you should be able to add very simple hyperlink navigation functionality to

your existing applications, documents, and objects. The Simple Hyperlink Navigation APIs hide all of that OLE interface stuff, making your entry into ActiveX hyperlinking quick and painless. But what if you wanted to do more advanced hyperlinking? What is actually going on underneath these Simple Hyperlink Navigation APIs?

You may want to incorporate more advanced features of hyperlinking into your applications. Authoring tools and browsers, for example, will need to implement a number of the ActiveX hyperlinking interfaces to host document objects. Five OLE interfaces make up the hyperlink interfaces: IHlink, IHlinkTarget, IHlinkFrame, IHlinkSite, and IBrowseContext (see **Figure 7**).

{ewc mvimg, mvimage,!illust.bmp}
Figure 7. Hyperlink OLE Interfaces

The ActiveX hyperlinking architecture centers around the Hlink object. Hlink is a standard system object that exposes an IHlink interface and encapsulates all of the information needed to hyperlink. This includes a target moniker (which points to where someone wants to go), a string that describes the location within the target, and some additional arguments, including a friendly name. In addition, the ability to navigate on behalf of the container is encapsulated in this object. This is accomplished through its IHlink::Navigate method. The Simple Hyperlinking Navigation APIs that I discussed earlier create an Hlink object from the arguments passed to the functions. They use this object to drive the navigation.

The Hlink object can be created through several APIs documented in HLINK.H: HlinkCreateFromMoniker, HlinkCreateFromString, HlinkCreateFromData, and HlinkQueryCreateFromData. The Hlink object can also be persisted because it imple-ments an IPersistStream interface. Since you can create an Hlink object from a data object, you can drag, drop, and paste it to the clipboard. The most important item encapsulated in the Hlink object is the moniker that points to a hyperlink target. A hyperlink target is the document to which you are navigating: an ActiveX doc object hosted in frame, a standalone application, an OLE object embedded in another document, or anything that a moniker can point to.

If you are implementing a hyperlink target, you have the option of implementing and exposing an IHlinkTarget interface. If you decide not to, users can still hyperlink to your object, but functionality will be limited. Generally, objects that support the IHlinkTarget interface are given information about navigating within a document. After the object has been instantiated through the IHlinkTarget interface, the hyperlinking framework can tell the object “here is the location within the document I want you to hyperlink to.” An example is hyperlinking to a Microsoft Excel spreadsheet and navigating to a specific cell within the spreadsheet. Another use of the IHlinkTarget interface is to pass a pointer to the standard system browse context object to the hyperlink target. The browse context object implements an IBrowseContext interface and knows the order in which documents were visited. All jumps are recorded with this context, and this context chains them together in a navigation stack, so it knows where to go in response to Go Back or Go Forward. In IE 3.0, the drop-down combo box under the toolbar displays the history list. Given the pointer to the browse context object’s interface, your application can enumerate and display the navigation stack.

Now that I have defined a target, I have to define the source of a hyperlink. The source is the hyperlink container — the object from which you are linking. The hyperlink container is responsible for the visual representation of the hyperlink, generally underlined text or a bitmap. You can represent a hyperlink any way you want — it could be a toolbar button or a menu — but you should try to be consistent with common methods of representation. (Microsoft is working on producing UI guidelines for representing hyperlinks.) An example of a hyperlink container is an HTML document. Generally, an HTML document displays underlined text that a user can select, which results in a hyperlink to another HTML document. What about selecting some underlined text within an HTML document and hyperlinking to a position in the same HTML document? In this case, not only is the HTML document the hyperlink container, it’s also a hyperlink target (see **Figure 7**). When I modified the BINDSCRB application to call HlinkSimpleNavigateToString when the user selects the Microsoft Home page, my application became a hyperlink container.

You don’t have to implement any OLE interfaces to become a hyperlink container, but if you implement an IHlinkSite interface, your hyperlink container will have access to additional information and your hyperlinking can become more efficient. A hyperlink container implementing an IHlinkSite interface will have access to all of the information contained within the IHlink object. In addition, when a hyperlink container calls a hyperlink function to jump to a hyperlink target, the container will be through the IHlinkSite::OnNavigationComplete member function notified when the hyperlink has been completed. To make navigating even simpler, if your document supports a base URL and the IHlinkSite interface, your users can use relative URLs. When a user wants to hyperlink to a relative URL, the hyperlinking architecture will call back into the IHlinkSite interface to obtain the base URL to resolve where it has to go.

The last interface in the hyperlinking architecture is IHlinkFrame. The container that hosts a document and implements the IHlinkFrame interface is called a hyperlink frame. An example of a hyperlink frame is IE 3.0 (see **Figure 7**). Through this interface, the hyperlink frame is notified when a hyperlink container has navigated to a new hyperlink target. From this information, the hyperlink frame displays the new document. In addition, the

hyperlink frame generally contains the user interface that displays the navigation stack and buttons or menu items to move backwards or forwards in the navigation stack.

The latest addition to the Win32 ActiveX extensions API is the URL Open Stream (UOS) functions (see **Figure 8**). These functions were made public in the ActiveX SDK beta posted on Microsoft's Internet site in June.

Figure 8. URL Open Stream Functions

Function	Description
URLOpenStream	Creates a push-type stream object from a URL
URLOpenBlockingStream	Creates a blocking-type stream object from a URL
URLDownloadToFile	Downloads bits from the Internet and saves them to a file
URLOpenPullStream	Creates a pull-type stream object from a URL
URLOpenHttpStream	Advanced function for doing more sophisticated HTTP and FTP downloads, such as performing an HTTP POST

The UOS functions are the easiest and most powerful way to download data from the Internet into your applications. These functions combine the familiarity of C programming with the power of COM. In fact, that's exactly what's underneath these functions. They use the services of URL monikers and WinInet, which is the ActiveX API set that lets you get information from the Internet into your application without using a browser (*see Nancy Nicolaisen's article on page 69 of this issue for more* — Ed). This means you get all of the caching and thread-synchronization features of these services every time you call the UOS functions. If you're calling these functions from an ActiveX container, the UOS functions will handle all of the binding operations. If you're not calling them from an ActiveX container, you won't be shortchanged; the UOS functions work equally well inside the ActiveX framework or within a standalone application.

Before jumping into the UOS functions let's cover a few basics — using these functions requires the knowledge and use of the IStream and IBindStatusCallback interfaces.

The IStream interface has been around for quite some time. This interface supports reading and writing data to stream objects using methods similar to the MS-DOS FAT file functions. For example, each stream object has its own access rights and a seek pointer. The main difference between a stream object and an MS-DOS file is that streams are not opened with a file handle. Instead, you use the IStream interface pointer. The methods defined for this interface present your object data as a contiguous sequence of bytes that you can read or write. How is the IStream interface related to the UOS functions? When calling the UOS functions, the data you request will be returned to you in a STGMEDIUM structure (a generalized global memory handle used for data transfer operations). The union member pstm specifies an IStream instance; you use the methods of the IStream interface to read the data.

IBindStatusCallback (see **Figure 9**) is a new ActiveX interface for asynchronous monikers. Since URL monikers are an implementation of asynchronous monikers and the UOS functions use the services of URL monikers, you will have to implement this interface when using a few of the UOS functions. **Figure 10** is a decision table that describes when to implement an IBindStatusCallback interface.

Figure 9. IBindStatusCallback Interface

```
interface IBindStatusCallback: IUnknown
    {
    HRESULT GetBindInfo([out] DWORD* pgrfBINDF,
                        [in, out] BINDINFO* pbindinfo);
    HRESULT OnStartBinding([in] DWORD dwReserved, [in] IBinding* pbinding);
    HRESULT GetPriority([out] LONG* pnPriority);
    HRESULT OnProgress([in] ULONG ulProgress, [in] ULONG ulProgressMax,
                       [in] ULONG ulStatusCode, [in] LPCWSTR szStatusText);
    HRESULT OnDataAvailable([in] DWORD grfBSC, [in] DWORD dwSize, [in]
                            FORMATETC* pformatetc, [in] STGMEDIUM*   pstgmed);
    HRESULT OnObjectAvailable( [in] REFIID riid, [in] IUnknown *punk);
    HRESULT OnLowResource([in] DWORD dwReserved);
    HRESULT OnStopBinding([in] HRESULT hrStatus, [in] LPCWSTR szStatusText);
    };
```

Figure 10. IBindStatusCallback Implementation

Function	Implementation
URLOpenStream	Mandatory
URLOpenBlockingStream	Optional
URLDownloadToFile	Optional
URLOpenPullStream	Mandatory
URLOpenHttpStream	Optional

To make things easy, you only have to implement the IBindStatusCallback::OnDataAvailable member function for functions that require an IBindStatusCallback interface. URLOpenStream and URLOpenPullStream call IBindStatusCallback::OnDataAvailable every time data arrives from the Internet. To abort the download, return E_ABORT from the OnDataAvailable call. Since the rest on the member functions are optional, you can simply return NOERROR or E_NOTIMPL.

If the implementation of the IBindStatusCallback interface is optional, you only have to implement the IBindStatusCallback::OnProgress member function. URLOpenBlockingStream and URLDownloadToFile call IBindStatusCallback::OnProgress on some connection activity. By implementing IBindStatusCallback::OnProgress, you can implement other progress-monitoring functionality like the progress bar in the IE 3.0 status bar. In addition, you can cancel the download operation by returning E_ABORT from the IBindStatusCallback:: OnProgress call. Once again, you can return NOERROR or E_NOTIMPL for the other member functions.

In all cases, IBindStatusCallback::GetBindInfo is never invoked for clients using the UOS functions. This is because the bind information is determined according to the UOS function being called. Let's take a look at each of the functions in more detail.

The URLOpenStream function creates a push-type stream object from a URL.

```
URLOpenStream (LPUNKNOWN pCaller, LPCWSTR szURL,
               DWORD dwResv,
               LPBINDSTATUSCALLBACK lpfnCB);
```

In a push-type data model, the URL moniker drives the bind operation and continuously notifies the client through the IBindStatusCallback::OnDataAvailable member function whenever data is available. Data is downloaded as fast as possible in this model.

The first argument to this function, pCaller, is a pointer to the controlling IUnknown of the ActiveX component. If the caller is not an ActiveX component, this value may be set to NULL. szURL is the string representation of the URL that will be converted into a stream object by the function, such as http://www.acmewidgets.com/giant_hair_dryer.gif. dwResv is reserved for future use, and lpfnCB is a pointer to the caller's IBindStatusCallback interface.

When a callback is invoked, the IBindStatusCallback::OnDataAvailable member function is called. A typical implementation of IBindStatusCallback::OnDataAvailable as used by the URLOpenStream function is shown in **Figure 11**. If the pstm member of the STGMEDIUM structure is not NULL, you can read from the stream the amount of data specified in the dwSize argument passed to the IBindStatusCallback::OnDataAvailable call. When the OnDataAvailable argument grfBSCF indicates BINDF_LASTDATANOTIFICATION, data will no longer be downloaded.

Figure 11. PushCBindStatusCallback::OnDataAvailable

```
HRESULT CBindStatusCallback::OnDataAvailable
(DWORD grfBSCF, DWORD dwSize, FORMATETC* pfmtetc, STGMEDIUM* pstgmed)
{
.
.
.
    if (dwSize < sizeof(BITMAPINFOHEADER) )
```

```
        return (NOERROR);  // not enough has been read yet

    // if we did not get the header information, read it now
    if (!g_bGotInfoHeader)
    {
        if (pstgmed->pstm != NULL)
        {
            DWORD dwRead;
            HRESULT hr = pstgmed->pstm->Read (&bmih, sizeof(bmih), &dwRead);

            if (SUCCEEDED(hr))
            {
                g_bGotInfoHeader = TRUE;
                return(hr);
            }
        }
    }
.
.
.
}
```

```
The URLOpenBlockingStream function creates a blocking-type stream object from a
URL.
URLOpenBlockingStream (LPUNKNOWN pCaller,
                       LPCWSTR szURL,
                       LPSTREAM *ppStream,
                       DWORD dwResv,
                       LPBINDSTATUSCALLBACK lpfnCB);
```

In this model, data is downloaded from the Internet on demand by a call to IStream::Read. When calling IStream::Read, your application or object will block until enough data has arrived. When you're reading the data, the URLOpenBlockingStream function will always try to obtain the bits from the local cache. If the data is not in the local cache, the function will try to put the downloaded bits into the cache so your application won't have to get the data from the Internet. This scheme provides a quick and efficient method of getting your data.

The pointer to the IStream interface, ppStream, is created and returned to you when you call the URLOpenBlockingStream function. As soon as you have a valid IStream pointer, you can begin to read from the stream object.

```
IStream * pStream;
URLOpenStream (NULL, L"http://www.msn.com/", &pStream,
               0, 0);

char buffer[0x100];

DWORD dwGot;
HRESULT hr = NOERROR;

do {
    hr = pStream->Read (buffer, sizeof(buffer), &dwGot);

    //.. do something with contents of buffer ...
    } while (SUCCEEDED(hr));
.
.
.
```

The URLDownloadToFile function downloads the bits from the Internet and saves them to a file that you specify in the szFileName argument.

```
URLDownloadToFile(LPUNKNOWN pCaller, LPCWSTR szURL,
                  LPCTSTR szFileName, DWORD dwResv,
                  LPBINDSTATUSCALLBACK lpfnCB);
```

If you decide to implement the IBindStatusCallback::OnProgress member function, you will get notified of the download progress. I show you how this works later.

The URLOpenPullStream function creates a pull-type stream object from a URL.

```
URLOpenPullStream (LPUNKNOWN pCaller, LPCWSTR szURL,
                   DWORD dwResv,
                   LPBINDSTATUSCALLBACK lpfnCB);
```

In a data pull-type model, the client drives the operation. You can control how much data you want to pull from the server and when. This is handy when you have a list box full of data to fill or a document containing several pages worth of data. Initially, you retrieve what you need to fill the display. When the user scrolls down in the list box or wants to display a new page in the document, your application or control can read more data. The data is downloaded on demand by calling the IStream::Read member function. If enough data is not available locally to satisfy the requests, the IStream::Read member function will not block. Instead, IStream::Read immediately returns E_PENDING and the URLOpenPullStream function requests the next packet of data from the Internet server. **Figure 12** shows a typical implementation of OnDataAvailable as it is used by the URLOpenPullStream function.

Figure 12. PullCBindStatusCallback::OnDataAvailable

```
HRESULT CBindStatusCallback::OnDataAvailable (DWORD grfBSCF, DWORD dwSize,
                                              FORMATETC* pfmtetc,
                                              STGMEDIUM* pstgmed)
{
    HRESULT hr = NOERROR;
    DWORD dwAmountToRead = dwSize - m_readSoFar;
    BYTE *buffer = new BYTE [dwAmountToRead];

    while (TRUE)
    {
        DWORD dwRead;

        hr = pstgmed->pstrm->Read (buffer, dwAmountToRead, &dwRead);

        if (hr = = E_PENDING)
        {
            // we'll get notified again when more data comes
            return (NOERROR);
        }

        if (SUCCEEDED(hr))
        {
            // ok, process bits .... and keep looping
        }
        else
        {
            // we have an error...
            return(hr);
        }
    }
}
```

URLOpenHttpStream is a catch-all function for doing more sophisticated HTTP and FTP downloads, such as performing an HTTP POST.

```
URLOpenHttpStream (LPUOSHTTPINFO *lphttpInfo);
```

By filling in a UOSHTTPINFO structure, you can explicitly tell the function the functionality you expect and how you want the bits delivered from the Internet. The structure comes from the UOS specification (see **Figure 13**).

Figure 13. Members of the UOS Structure

Member	Description
ULONG ulSize	Size of this structure.
LPUNKNOWN punkCaller	Pointer to the controlling IUnknown of the calling ActiveX component (if the caller is an ActiveX component). If the caller is not an ActiveX component, this value may be set to NULL. Otherwise, the caller is a COM object that is contained in another component (such as an ActiveX control in the context of an HTML page). The argument represents the outermost IUnknown of the calling component. The function will attempt the download in the context of the ActiveX client framework and allow the caller's container to receive callbacks on the progress of the download.
LPCTSTR szURL	URL to be downloaded.
LPCTSTR szVerb	GET, PUT, POST, or a custom verb understood by the server. If NULL, GET is assumed.
LPCTSTR szHeaders	HTTP headers to use during connection to server. Can be NULL.
LPBYTE szPostData	Additional data to send after the headers. Typically this will be a POST data arguments. Can be NULL.
ULONG ulPostDataLen	Size of szPostData. Must not be 0 if szPostData is not NULL.
ULONG fURLEncode	Flags with either the UOS_URLENCODPOSTDATA, UOS_URLENCODEURL, or can be NULL. The two flags can be combined by a bitwise OR.
ULONG ulMode	Can be one of UOSM_PUSH, UOSM_PULL, UOS_BLOCK, or UOS_FILE. Each of these maps to one of the functions above and makes this function's programming model fit the corresponding function.
LPCTSTR szFileName	Name of the local file to write URL data to if ulMode is UOS_FILE. Otherwise, must be NULL.
LPSTREAM *ppStream	Pointer to IStream pointer if ulMode is UOS_BLOCK. Otherwise, must be NULL.
LPBINDSTATUSCALL BACK lpbscb	Pointer to IBindStatusCallback interface. This interface and the caller's programming model responsibility depend on the ulMode flag above. The function will behave exactly like the corresponding UOS functions above, depending on the flag settings.

The following sample application illustrates how the URLDownloadToFile function downloads the bits of a file, represented as a URL, from the Internet to a file created on your local machine. I call the sample application "UOS AVI Downloader" (see **Figure 14**) because it downloads an AVI file from the Internet and plays the AVI file within the Media Architects Video Play OCX control distributed with Visual C++ 4.1. You will need this OCX control to run this application.

{ewc mvimg, mvimage,!illust.bmp}
Figure 14. UOS AVI Downloader

The UOS AVI Downloader is a dialog-based MFC application. It contains an edit control where you enter the URL of the AVI file you want to download, a Retrieve button that you select to download the file, and the Media Architects Video Play OCX control. If you want to build the application, you will need Visual C++ 4.1 and the URLMON.LIB, URLHLINK.LIB, WININET.LIB, and UUID3.LIB libraries from the ActiveX Beta SDK. (The names of these libraries may be different when the final ActiveX SDK is released.)

All of the work is done in the BN_CLICKED message handler for the Retrieve button (see **Figure 15**). In the message handler, I create an IBindStatusCallback object, which I implemented in DOWNLOAD.CPP. I want to know when the download is complete, so I implement the IBindStatusCallback::OnProgress method. In this method I set a private data member m_gotFile to TRUE when the ulStatusCode argument is equal to BINDSTATUS_ENDDOWNLOADDATA. The asynchronous moniker specification lists all of the status codes returned to the ulStatusCode argument. By receiving different status codes during the download process, you can give your user feedback on the progress of the download.

Figure 15. AVIDNLDR Retrieve Button Message Handler

```
////////////////////////////////////////////////////////////////
// CAVIDownloaderDlg::OnBtnRetrieve - Message handler for the Retrieve
// command button.  Downloads the .AVI file from the Internet and runs // it in
the Media Architects Video Play OCX Control.
//
////////////////////////////////////////////////////////////////
void CAVIDownloaderDlg::OnBtnRetrieve()
{
    // This application does not do any checking to insure that the
    // user has entered a valid URL.  This exercise is left to the
    // reader.

    // Create a IBindStatusCallback object
    ptrURLDownloadToFileCallback pURLDownloadToFileCallback;

    if (!pURLDownloadToFileCallback)
    {
        AfxMessageBox (_T("Unable to create IBindStatusCallback object."));
        return;
    }

    // Disable the "Retrieve" button
    m_btnRetrieve.EnableWindow(FALSE);
    BeginWaitCursor();

    // Create a temporary file to hold the the .AVI file.  This demo
    // does not manage files.
    LPTSTR lpTempFileName = new TCHAR[MAX_PATH];
    LPTSTR lpPathName= new TCHAR[MAX_PATH];

    // Retrieve the path of the directory designated for temporary
    // files.
    ::GetTempPath(MAX_PATH,lpPathName);

    // Create a name for a temporary file.
```

```
    if (0 != GetTempFileName (lpPathName, _T("AVI"), 0,
           lpTempFileName))
    {
        // Swap out the .TMP extension with .AVI
        CString szFileName = lpTempFileName;
        szFileName = szFileName.Left ((szFileName.GetLength() - 3)) +
              _T("AVI");

        // Call the UOS function to do the file download
        CString szURL;
        m_URLString.GetWindowText (szURL);
        HRESULT hr = URLDownloadToFile (NULL, szURL, szFileName, 0,
              pURLDownloadToFileCallback);

        if ((hr = = S_OK) && (pURLDownloadToFileCallback -> got_File ()
               = = TRUE))
        {
            // Setup the Media Architects Video Play OCX Control
            // and play the .AVI file that was downloaded.
            m_VideoPlay.SetFilename (szFileName);
            m_VideoPlay.SetLoop (TRUE);

           // Play the entire video
           VARIANT varEntireVideo;
           varEntireVideo.vt = VT_ERROR;
           m_VideoPlay.Play (varEntireVideo, varEntireVideo);
        }
    }

    delete [] lpTempFileName;
    delete [] lpPathName;

    pURLDownloadToFileCallback -> Release();

    m_btnRetrieve.EnableWindow(TRUE);
    EndWaitCursor();
}
```

After the IBindStatusCallback object has been created, I create a temporary filename to hold the downloaded AVI file in the directory designated for temporary files. GetTempFileName creates the file with a TMP extension. I change the extension to AVI before calling URLDownloadToFile because the Media Architects Video Play OCX control will only play files with the AVI extension. I then call URLDownloadToFile passing NULL in the first argument (because I am not calling the function from an ActiveX component), the URL entered in the edit control, the temporary AVI file where I want the bits placed, and a pointer to my IBindStatusCallback object. The URLDownloadToFile function blocks until the download operation is completed. On a separate thread, the function will be calling my IBindStatusCallback::OnProgress member function to provide the download status.

When the URLDownloadToFile function completes, I check its return value and call the CBindStatusCallback member function got_File (which I created), to determine if the application actually received the file. If all goes well, I set a few properties and call some methods implemented by the Media Architects Video Play OCX control and the AVI file is displayed and playing in the UOS AVI Downloader application. In **Figure 14**, I downloaded the CUP.AVI file from http://www.microsoft.com/ie/avi. This cup is used in Microsoft's Volcano Coffee HTML demo page.

If you're writing ActiveX documents (see "The Visual Programmer Puts ActiveX Document Objects Through Their Paces" by Joshua Trupin, *MSJ* June 1996), you will get the integrated navigation user interface because IE 3.0 is capable of hosting ActiveX documents in-frame. Since your ActiveX documents are in-frame, you will get the in-frame hyperlinking navigation for free. If you are not writing ActiveX documents or you want to add hyperlinking navigation to your applications or OLE controls that you have developed today, the easiest way is to use the Simple Hyperlink Navigation APIs.

Just as it's important to navigate to documents, it's also important to access data as it moves to servers with Internet access. Using the URL Open Stream Functions you can retrieve data from these servers quickly and easily. You also have a large choice in the type of data retrieval model, data-push model, data-pull, and blocking model.

Click here to connect to the Macmillan Computer Publishing Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Macmillan Computer Publishing, a division of Simon & Schuster, is the world's largest computer book publisher. Written by leading experts in the field, Macmillan Computer Publishing publishes over 1,200 easy to understand books on all the major computing topics each year. Their Web site contains up-to-date references that are a comprehensive resource for the computing enthusiast.

Before jumping in on the thin client application development model, let's review why everybody is so excited about using the Internet infrastructure for application deployment. The most obvious reason the Internet is so interesting is because it gives users remote access to information. More information is available than will ever fit on a client machine. The question then becomes, which model best enables on-demand, source-independent information browsing?

Figure 1 shows the basic client/server model of the Internet. The client provides the presentation context, typically a browser such as Microsoft Internet Explorer or Netscape Navigator that allows remote data to be viewed through the HTML presentation layer. This data represents a slice of the content of the Internet. The network moves the content to the presentation context via HTTP.

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. Client/Server Model

URLs further separate the location of content from the presentation context, allowing a browser to access multiple servers and their associated data. Actual access to the content is deferred until the user wants to see it. This deferral mechanism places minimal demands on the network and avoids the transfer of unnecessary data to the client machine. This approach facilitates the exposure of a maximum amount of remote resources with the minimum use of network and client-side resources (see Figure 2).

{ewc mvimg, mvimage,!illust.bmp}
Figure 2. Remote Data

Remote data isn't the only reason why the Internet is interesting — users also want to access logic and services not on their local machines. URLs are as useful as locators for remote logic and services as they are for remote data. It's the combination of data and logic that forms the new content for the Internet. As illustrated in Figure 3, the structure is the same.

{ewc mvimg, mvimage,!illust.bmp}
Figure 3. Remote Data and Logic

The new content for the Internet is objects — specifically, C++ classes that exist on the server. These are the regular kinds of C++ classes you create, edit, compile, and debug with the Visual C++ development environment. Although the current approach to Internet deployment is typically procedural, where the data operates separately from the logic, it can be enhanced with the object-based model you're used to.

ActiveX and Java give application deployment more flexibility. The client becomes a mechanism for accessing remote objects on remote servers (see Figure 4). The server contains data and logical content in program structures such as C++ classes.

{ewc mvimg, mvimage,!illust.bmp}
Figure 4. Remote Object Deployment

How can remote objects best be exposed to the client user over the network infrastructure? The Internet's greatest potential is realized when logic and services, together with content, are presented to multiple users in a context of transparent, real-time collaboration (see Figure 5). Collaboration in a network environment means that two or more users can communicate with one another by viewing or interacting with common data and services. The sessionless nature of the Internet lets multiple clients access an object's state concurrently.

{ewc mvimg, mvimage,!illust.bmp}
Figure 5. Real-time Multiple User Collaboration

Collaboration on the Internet removes the user's dependency on the location of information and services by allowing many users to interact with information freed from the constraints of monolithic applications. But collaboration in an Internet environment is a particularly difficult task.

So you're convinced that you want to take advantage of the Internet for deploying applications. How can you be sure that the tools you select or the methods you employ for building and publishing your applications are going to yield the best results? Let's develop a litmus test to ensure that the solution you choose will meet your objectives. When building applications for the Internet, ask yourself the following questions: What architectural features mesh most logically with the structure of the Internet? What are the requirements for a development framework to create Internet object-based applications using Visual C++? Why didn't I become an accountant? How can I best use an Internet client/server architecture with the Visual C++ development environment I already use? What is a good Internet application development solution, not just for the Internet of today but into the future?

Let's take a brief look at some of the features you need to get your C++ objects to the Web.

An important part of the litmus test is the application-deployment mechanism. Applications should be able to run without needing client software or plug-ins to be installed prior to connection. This dynamic deployment mechanism enables all or part of the application to be downloaded to the client machine and run locally, allowing the application to benefit from local machine services such as the display, input devices, and local interactive controls in the user interface. Application performance and interactivity are made possible because code is downloaded and run on the local machine.

What parts of the application really need to be downloaded and run? Certainly not the application logic. Application logic interacts with data not on the client machine (for example, corporate databases, data repositories, and shared information services). In addition, the application logic I'm talking about consists of C++ objects running on the server.

The part of the application that gets downloaded and run on the client machine is the presentation layer. But how do you get the presentation onto the client machine? This presentation layer includes the GUI elements along with just enough application logic to convert user interface events into object state changes. These state changes, and only these state changes, are what your client will send across the Internet to the server.

One way to do this is by using what I call "Just Enough Java." This use of Java is simply pragmatic. Besides providing a dynamic download of the presentation layer, Java offers platform independence, effectively making your Visual C++ applications cross-platform. In addition, as a client-side solution, Java takes advantage of its virtual-machine security model.

Thinner is better. No, this isn't an infomercial for the Thighmaster. When it comes to Internet clients, thin is definitely in. The interesting thing about the Internet is that most of the data and logic that users want to access is not on their machines. What do you send down to the client machine to successfully access and deploy an application? Here, less is more. Why? Let's review the fat client problem.

Practically all of the work done with Java to date uses a fat client model. That is, the application logic, not just the presentation layer, is downloaded to the client machine for execution. Why is this a bad approach? First of all, by downloading the application logic to the client, the logic is displaced from the content the user is trying to access. This violates the basic object-oriented model of data/logic encapsulation. The application has to be split, code has to be sent down to the client, and a network protocol has to be created and maintained by the developer. The whole process involves creating a distributed application, greatly increasing the complexity of writing an application simply to move it to the Internet.

Another problem with fat clients is the burden that the application places on the client. If too much of the processing burden takes place on the client machine or device, fewer types of client applications and platforms will be able to access and run that application. This will become a problem when Internet appliances become popular. Cramming a large application into a narrow-bandwidth pipe and into a limited-performance client device inhibits your ability to move your application to the net. The fat client approach is not appropriate when you cannot rely on the power or capability of the client. So, if you've got visions of being able to access your Visual C++ application on a Sega Genesis machine, a cellular phone, or a PDA device, remember: the thinner the client, the fewer restrictions on deploying your application.

Fat clients also suffer because of the bandwidth limitations of the Internet. Since the application logic is on the client in a fat client approach, large amounts of data must be moved to the client machine before the application is able to process it for the user. This movement of data puts an unnecessary bandwidth burden on the network connection. This is problematic because the Internet is largely a low-bandwidth, high-latency environment not conducive to large amounts of network traffic.

Finally, the fat client inhibits collaboration. It becomes a burden to replicate a common state across multiple machines when the application object state is controlled in a distributed fashion. In other words, if the client contains logic that changes a shared object's state, that state must somehow be updated on other clients currently looking at or interacting with that object. This becomes a tremendous burden on the developer. The problem can be remedied somewhat through an RPC or object proxy solution, where multiple clients are accessing centralized data at the server. But client synchronization and data access control are typically left to the developer, since RPC-like mechanisms don't deal with the replication issues associated with collaborative applications.

RPC approaches also include the burden of deciding where to split the application between the client and the server. The developer must write a distributed application where some of the code is Java, JavaScript, or VBScript and the rest is a server object written in C++. Doesn't sound like a very scalable development solution, does it? Besides having to decide which parts of the application are where, the developer has to make coding decisions about what data is changed, whether it is shared or not, and so on. Some technologies attempt to abstract the distributed nature of the application, often requiring the use of a proprietary language, generally resulting in a loss of control over performance issues.

When it comes to deploying applications with the Internet architecture, thinner is better because it allows you to distribute the presentation, not the application. Rapid application access is possible since you do not have to download the entire application. A minimal burden is placed on client resources (or smaller dependency on hardware configuration), and development complexity is reduced because you don't have to write a distributed application. So the litmus test includes a check for thin client Internet application deployment.

Another issue for the litmus test is in the area of collaboration. Collaboration should be a central theme when you talk about the Internet. A Web development effort should not be considered without taking into account how multiple simultaneous users will access that application. Any environment or framework should be able to abstract or automate collaborative capability at some level.

I believe that there are two ways in which collaboration can be made possible with minimal developer involvement. The first would be through a codeless presentation model. By codeless I mean that the presentation layer of an application would contain as little application logic as possible (or none) that directly manipulates shared object states. What this means is that object state manipulation code shouldn't be downloaded to the client machine. In an ideal codeless presentation model, the presentation logic executed on the client machine deals only with user interaction (GUI events), transforming those events into data state change commands. Client data state changes should then be packaged and sent to the server to update its objects' states and, in turn, update any other interested clients currently looking at the state of those server objects. The logic that operates on the shared data would be an encapsulated C++ object running on the server machine. The presentation layer would represent a remote client to that object's services. The central shared object would be accessed by multiple simultaneous clients.

Another collaboration technique allows many clients to have simultaneous dynamic access to a running application. An application framework with such a runtime binding capability could allow clients to attach or detach a particular application during program execution. Since application logic is just another kind of content that can be encountered on the Internet, the same should be the case for applications as well. Runtime binding on an application would allow a client to access an application object's state without involving the shared object directly. By dynamically allowing access to the C++ object, the details for client attachment to that object wouldn't have to be handled by the implementor of the class. This general lack of direct developer involvement in the control of collaboration access would greatly simplify the process of writing a multiuser-access Internet app.

Available bandwidth has always been a limiting factor for effective deployment of applications over the network, so the litmus test should include an examination of the bandwidth impact.

Where do you divide an application to ensure that the minimum amount of presentation logic is sent to the client and that the application runs optimally over the net? The division shouldn't take place at the event level, since many more events occur on the client than are necessary to maintain an application object's state. Passing events from one side to the other causes the application to create many data round trips before users see the results of their interaction on the screen.

For example, many events happen with a scroll bar interactor, but the important piece of information is the thumb position produced through user interaction. If the scroll bar controls the value of a numeric data member of a C++ object, then only the final scroll bar thumb position value needs to be sent across the network to update the server-side object. This transfer of state information is the minimum information exchange necessary to maintain synchronized application state over a network. To minimize state-transfer bandwidth, you need to send only that part of an application object's semantic state that is currently relevant to the presentation. A core question of GUI research over the last decade has been how best to implement the state transfer.

A model is needed that utilizes the state-based nature intrinsic to object-oriented devel-op-ment. Such a model is essential for optimal communication between the server application and the many clients that can simultaneously access that application. Ideally, the implementation of C++ objects need not be involved directly with this state-change notification scheme. Object state management would then remain external to the application object in the same way a client would access the services of an object without the object being concerned with the details of how it is being used. This approach is appealing because it is not cluttered with unnecessary dependencies such as how the object is used, deployed, accessed, and so on.

The litmus test just created effectively defines a new application architecture for the Internet. This model, which I call the thin client three-tier architecture, can be differentiated from the traditional three-tier client/server approach in two ways. First of all, in the thin client model, all the application-specific logic is running on the server. The logic could be distributed on the server side over many machines, but the approach is inherently server-side. Secondly, in the thin client model, only state changes are exchanged between the client and the server. In the traditional three-tier model the communication between the client and the server is based on remote procedure calls (RPCs). Implementing an application with RPCs requires application-specific logic on the client. Application partitioning is typically left to the developer. The problem with the traditional three-tier architecture with respect to the Internet is that it exhibits many fat client characteristics. It is important to note that a thin client approach allows objects to be accessed by multiple clients simultaneously and in real time.

Traditional client/server approaches, allowing only a single client to access a server process, were OK for the single-user desktop application, but inadequate and too inflexible given the multiclient access capabilities of the Web. Traditionally, multiuser access to data was limited to the multiuser capabilities of the database engine. The thin client approach enables multiuser control of data and services of application objects. The traditional client/server approach controls data and services at the level of the database engine where the context of the data with respect to its associated logic is limited or nonexistent.

Let's take a look at the implementation of the thin client three-tier architecture. Client, server, and database pieces can be interchangeable. The Java AWT toolkit provides the API for the user interface of an application. The presentation layer (the thin client) can actually be an applet downloaded from the server and run within an Internet browser. The applet contains user-interface widgets such as scroll bars and buttons and enough embedded network communication machinery to respond to and initiate state changes between the client and the server.

A client-side visual element is associated with a specific server-side object. For example, an AWT edit box on the client can be "connected" to a data member of a C++ object running on the server. When a value is typed into the edit box on the client-side applet, a data state change message is sent to the server. In this example, an edit box is a Java applet with a "network" line connected to a server-side object's data member. Any other clients viewing that data member through their interfaces will also update automatically to reflect the object's new state.

Although the client is small, it is also smart enough to know the kind of C++ objects running on the server to which it is connected. The client sends network messages when the user interacts with the AWT control, and receives state changes initiated by the server (for example, when the application logic changes the data member). The generated presentation layer then becomes a client to the server-side C++ objects. The entire client can be generated so that, by default, the developer isn't required to write any code (see Figure 6).

{ewc mvimg, mvimage,!illust.bmp}
Figure 6. Thin Client Three-tier Architecture

C++ objects are executed on the server. The client download initiates a connection to server objects through an object framework-based protocol. Clients can attach to an already running C++ application on the server or start a unique C++ process upon client-applet startup. Any database accessible through C++ can be accessed by the objects in the server-side application.

Having defined the requirements of Internet applications and the architecture necessary to implement them, let me highlight the specific characteristics of a Visual C++ development framework that would enable the creation, deployment, and maintenance of such applications. The framework includes the following capabilities and features: meta data, state change notification, callbacks, runtime binding, cycle reduction, synchronization, and model-view separation. Now, let's take a closer look at each of these components.

Meta data is a format for describing the structure of application objects. In the world of C++, meta data would include information about a class structure, for example, the name of a class, its functions and fields, their types, and so on. The Internet's protocol for discovering the structure of content on the server is HTML. The HTML format couples content and meta data for remote information access. HTTP makes requests on parts of the server content by performing a GET command. The argument of the GET command (the path to the HTML file of interest) is really information about the data the user is trying to access — the meta data. Ideally, application objects are viewed and manipulated over the Internet by accessing them through what I like to call a meta interface.

An effective framework for accessing C++ objects over the Web should be inherently self-describing. In other words, clients are able to access objects' services and structures to attach to these objects. A protocol similar to HTTP incorporated into the framework would enable applet clients to perform application-level GET commands for accessing server-side C++ objects. This same protocol would also support server-side pushes that move C++ object state changes to the client when the object state changed. Meta information is the glue that enables remote presentation to attach to C++ objects on the server.

While meta information provides remote access, a physical network connection must exist to communicate state changes between the client and the server. State change notification capability must be supported at the framework level. A framework that requires the developer to explicitly deal with data change notification messages adds too much complexity to the application code and places too many dependencies on the details of remote presentation access into the application logic, making application objects inherently nonreusable.

Callbacks are functions called when data changes. C++ provides hooks for allowing functions to be called when data changes in a way that is transparent to the implementor of a class. Suppose the following class represents a way to store and manipulate an *x*, *y* position:

```
class Point {
public:
 Point(int x, int y) : xPos(x), yPos(y) { }
 void offset(int x, int y) { xPos += x; yPos += y; }
 void setXPos(int x) { xPos = x; }
 void setYPos(int y) { yPos = y; }
 int getXPos() { return xPos; }
 int getYPos() { return yPos; }

private:
 int xPos;
 int yPos;
};
```

The xPos and yPos data members represent the actual information of the Point class. These data members can have “guarded” or write barrier access (an Eiffel language term) or they can be “computed fields” (as in Smalltalk). Each of these implementations fires a callback when data changes. Another technique, C++ operator overloading, allows code to be executed whenever an explicit assignment to the data member is performed.

Let’s first introduce a new class and then modify the Point class shown above to include write barrier capability. The following IntData class represents a partial encapsulation of the int type:

```
class IntData {
public:
 Int(int d) { data = d; }
 int operator () { return data; }
 operator = (IntData &d) {}
 operator = (int d) { }

 addCallback(void f(int));
 removeCallback(void f(int));
protected:
 dataChanged()
private:
 int data;
};
```

Now, let’s modify the Point class to have callback-enabled data members:

```
class Point {
public:
 Point(int x, int y) : xPos(x), yPos(y) { }
 void offset(int x, int y) { xPos += x; yPos += y; }
 void setXPos(int x) { xPos = x; }
 void setYPos(int y) { yPos = y; }
 int getXPos() { return xPos; }
 int getYPos() { return yPos; }

private:
 IntData xPos;  // CHANGED
 IntData yPos;  // CHANGED
};
```

Notice that the only code in the class that needed to change (as indicated with comments) is the declaration representation of the Point data members, using IntData as the type. Using this mechanism, when a Point

object's state changes, clients external to the class can be notified. You'll see a little later that client access to data members can be reasonably restricted to a limited set of clients, for example, clients that visually represent the object remotely. Not just anyone can access the internal state of an object.

This capability allows callbacks to be associated with the state of objects. For brevity, many of the details of encapsulated access have been left out. This method can be applied to C++ object pointers, arrays, reference-counted objects, and so on. This approach to data change notification contributes to a dramatic reduction in the complexity associated with connecting a remote thin client user interface to the object itself. Data members, then, use a "plug and socket" access approach that allows the representation of the object to be external to the implementation of the class.

Data change notification is not quite enough. Suppose that clients want to modify the object during program execution without generating code. Code generation complicates the task of attaching multiple heterogeneous clients to server objects. For maximum flexibility in attaching collaborative clients to an object over the Web, a runtime binding mechanism can be added to C++ object access. C++ runtime binding doesn't degrade the general performance of the running object itself. Remember the advantages of having meta data information available to describe the structure of classes? Well, it can also help automate the runtime binding of compiled C++ classes. The meta data of each class that is to be accessible over the Internet can be used to generate a C structure that enables lookup and access to data members of an object by name (for flexibility) and by index (for speed). Of course, the name can be resolved to an offset index at runtime as an optimization.

In Figure 7 the class represents color, which contains red, green, and blue components. Runtime binding access on member functions is also possible, which can further separate the client presentation from the structure of the object. The runtime binding approach allows you to preserve the performance capability of the C++ object itself while allowing remote dynamic access to the object. All this is possible without introducing appreciable complexity for the class implementor (since the runtime binding structure can be automatically generated using the meta data of the class). This lets you focus on the business logic without concern for the details of making the object's state externally viewable.

Figure 7. Using Meta Data with Runtime Binding

```
class Color {
public:
 Color(int r, int g, int b) : red(r), green(g), blue(b) {}
 int getRed() { return red; }
 int getGreen() { return green; }
 int getBlue() { return blue; }
 void setRed(int r) { red = r; }
 void setGreen(int g) { green = g; }
 void setBlue(int b) { blue = b; }

private:
 // "callback enabled" data members
 IntData red;
 IntData green;
 IntData blue;
};
```

When monitoring data changes on objects, cycles can occur that would cause infinite callback conditions. For example, the following doTransaction function gets called:

```
void doTransaction()
{
A = 10;
}
```

A callback function, A-Changed(int a), was added to the callback list on the A member so that, when A is assigned to 10, the AChanged function is called:

```
void AChanged(int a)
{
B = 20;
                      }
```

A callback function, BChanged, was in turn added to the B data member, and so it is called as well.

```
void BChanged(int b)
{
 doTransaction();
}
```

Notice that the BChanged function calls the same function that started the chain of functions to be called in the first place. This creates an infinite cycle. Unless reduced, this sequence of changes and notifications will happen at a furious pace so long as the system stack will permit. The solution is to set a flag so that if the data change callback notification happens while that same data is being changed, the callback function isn't called again.

Concurrency control always becomes an issue when dealing with multiple threads of execution. In the thin client three-tier model, multiple clients can update the state of shared objects on the server at the same time. How do you resolve synchronization problems as they occur without coding the concurrency control into the objects themselves? Keep in mind that simple proxy methodologies or RPC mechanisms don't handle concurrency issues automatically. A remote function call that changes an object's state can corrupt or break other clients making similar requests on the same object at the same time.

Let's look at a fairly simple example. Suppose a virtual whiteboard object is running on the server. It contains a list of shapes that have been drawn on the whiteboard by several remote thin clients. The current state of the whiteboard includes a square (item 0), triangle (item 1), and a line of text (item 2). Three clients are currently viewing the whiteboard. Client 1 invokes a command to delete the text (item 2). At the same time client 2 invokes a command to delete the triangle shape (item 1). Also at the same time, client 3 sends a command to add a line. Without concurrency control, the object state will depend on the order in which the commands are handled by the server object.

Here's an example of how optimistic concurrency control resolves this problem (see Figure 8). Let's say the server handles or sees the "delete the triangle" (item 1) first. The delete command is performed successfully. Next, the server processes the command to delete the text (item 2). Of course, by now the previous delete of the 0th item could corrupt the intent of deleting item 2. The second delete command needs its index to be decreased by one to index 1 (a transformation) to preserve the intent of the command (deleting the text item). Finally, the third command to add a line at the end of the shape list needs no modification since the index of the line item depends on what is currently the last item index. The key to making this scenario work correctly is to use an optimistic algorithm that preserves object states. It is an optimistic approach because no client-to-server negotiation takes place to resolve state change commands. The algorithm assumes that commands can be transformed as they are encountered to preserve the object's integrity. In Figure 8, the immediate action on the whiteboard is translated into the final resolution by the optimistic algorithm.

{ewc mvimg, mvimage,!illust.bmp}
Figure 8. Optimistic Concurrency Control

The concurrency control example given above is specific to array or list-based data structures. Each data type requires a unique set of rules appropriate to it. Typically, a temporal list is maintained to monitor and adjust operations on data structures based on the data type's rules. Commands from the clients can be transformed to preserve the integrity of the transaction. Again, this synchronization capability can exist outside of the regular implementation of application-specific C++ classes if runtime binding and data change callbacks are integral to the object structure or framework.

In the thin client architecture, user interfaces visually represent objects running on the server. In an ideal environment, the presentation is separated from the objects themselves. Figure 9 illustrates the point.

{ewc mvimg, mvimage,!illust.bmp}
Figure 9. Presentation Decoupling

In Figure 9**,** a class (ChangeValue) contains a single data member (value) and two functions (addToValue and subtractFromValue). This ClassView tool is part of the framework package available from the sources listed on page 5. The ChangeValue class, when instantiated at runtime, will execute on the server. A dialog can be created that visually represents the entire class. The scroll bar and the edit box are both client controls attached to the class member int value. The buttons control the direct invocation of the member functions of the class. The lines shown between the client presentation and the server C++ object represent the state change management relationship that exists between the thin client and the server-based application object.

In this example, the user would look at the presentation embedded in the HTML page as a Java applet. The object (the effective content for the client presentation) is manipulated as the user interacts with the thin client interface. For example, suppose the user presses the Add button. This causes the client to initiate a command to call the member function addToValue on the ChangeValue class. The following code is written for the addToValue function:

```
void ChangeValue::addToValue()
{
 // change the value, this may create callbacks to occurs depending
 // on the number of clients currently interested in this value
 value += 10;
}
```

The value change will produce a set of callbacks to be invoked, which will subsequently update any thin clients currently attached to the ChangeValue object, including the scroll bar and edit box shown on the page where the user pressed the button.

My thin client Internet sample application uses C++ code on the server and a Java applet running in Internet Explorer on the client. This basic chat program comprises only a few pages of code but manages to show the collaboration capability inherent in server-centric architectures. The program allows users to add messages to a chat listing that all attached clients can see in real time. Clients add to the common list by typing in the message and invoking a send command. Users can identify themselves via an edit string. Clients are dynamically attached to and detached from a running server-based chat application. While all clients share the common chat list, other information is client-specific and is invisible to other users. The application structure along with the Java client presentation is first defined in the Internet-enabled app builder, part of my framework (see Figure 10).

{ewc mvimg, mvimage,!illust.bmp}
Figure 10. Chat Applet in Development

The C++ class definitions are defined in the builder, which uses class schema information to build connections to the Java-based dialogs. As the class structure and the Java client dialogs are designed, the application developer invokes the commands to generate the basic pieces for deploying an Internet application. On the server side, basic C++ skeleton code is created by the builder (the .h and .cpp files for the classes defined in the builder). On the client side, a Java applet is generated and automatically compiled by the builder prior to deployment to the client in the context of an HTML page. As a convenience, a Visual C++ .mak file is generated so that the server side application compiles to an .exe. The application developer then implements the app-specific logic in the member functions of the classes and compiles the app directly. Code editing, debugging, and general development all take place in the familiar Visual C++ environment. On the client side, there is no more code to write. By default, an HTML file is generated for each project created in the application builder environment. The Java applet is simply referenced in an HTML file and automatically runs when the user double-clicks on the reference. Let's first look at the C++ code you would write.

Two classes are defined in the builder: ChatApp and ChatPerson. ChatApp is derived from my EosFramework, which provides basic application-level procedures. You can think of the EosFramework class as the main() of the server side application. It has functions that control various program aspects: collaboration control, application startup and shutdown, and so on. An array of strings is defined in the ChatApp class that represents all the messages sent by the various clients since the application started (see ChatApp::fChatList in Figure 11). The EosPrimitiveArray class is templated for string types and uses the MFC CArray class for its storage and basic functionality. The fChatList data member of ChatApp defines the data that is to be shared among all attached clients. Two other functions, getFirstObject and getClientViewName, are used for controlling multiple collaborative clients (see Figure 11).

Figure 11. chatApp.h

```
// chatApp.h
class ChatApp: public EosFramework
{
public:
ChatApp();
ChatApp(const ChatApp& src);
virtual ~ChatApp();
ChatApp& operator =(const ChatApp& src);

      // overriden functions from the base application framework class
      virtual EosObject  *getFirstObject();
      virtual EosAtom  getClientViewName(EosClient *client);

private:
// storage for the chat strings based on messages created by remote clients
      EosPrimitiveArray<EosString> fChatList;
};
```

The chat application is launched for the first time when a Java thin client applet requests access to the server-based application. Since this application is designed to be collaborative, each new client attaches to the same chat application process running on the server. The rules regarding client to server connections can be defined by the developer, including attaching a single server process to multiple clients or attaching a only one client to a server process. In the application example, I want each client that attaches to the chat process to receive its own private object. This object, ChatPerson, allows the user to view common data (the chat list) as well as its own uniquely viewed information. The function ChatApp::getFirstObject is called at runtime whenever a new client attaches to the server's running process. It returns the first object that will be presented to the user as a Java applet. The state of the data associated with the ChatPerson instance will be automatically reflected in the Java applet user interface based on the dialog that was designed for that object in the application builder. The function ChatApp::getClientViewName (see Figure 12) allows the developer to control exactly which dialog designed in the builder will be initially presented to the user on the client.

Figure 12. chatApp.cpp

```
// chatApp.cpp
#include "stdeos.hpp"
#include <chatApp.h>
#include <chatperson.h>

// retrieves a chat person object for each client that attaches to the server .exe
EosObject *ChatApp::getFirstObject()
{
  return (EosObject *)new ChatPerson(&fChatList);
}
// allows the server app to control the object and dialog name that is to be used
at the applet level
// of the remote presentation
EosAtom ChatApp::getClientViewName(EosClient *client)
{
```

```
  return EosAtom("ChatPerson.Chat");
}

// misc. member functions
ChatApp::ChatApp() : EosFramework(),
    fChatList()
{
}
ChatApp::ChatApp(const ChatApp& src) : EosFramework(src),
    fChatList(src.fChatList)
{
}
ChatApp::~ChatApp()
{
}
ChatApp& ChatApp::operator =(const ChatApp& src)
{
    if (this != &src)
    {
        EosFramework::operator=(src);
        fChatList = src.fChatList;
    }
    return *this;
}
```

The ChatPerson object will actually be presented to the client remotely in the Java applet. One instance of ChatPerson is created for each client that attaches to the chat application. When the client detaches from the server side application, its ChatPerson instance will be deleted as well.

The basic framework defined here allows the developer to define C++ object pointer members with a reference-counted template wrapper class. This class is useful for determining when an object is no longer interesting to other parties. This is especially useful in the case when clients attach and detach at runtime. When a client detaches from the server process, and if that client held the only reference to the object, it will automatically go out of scope. This greatly reduces object pointer management often associated with C++ based applications. The ChatPerson member fChatList is another example of automatic reference counting. A reference to the global chat list is assigned to the client in order for the ChatPerson object to attach client-specific messages. Only when the actual application terminates does the global chat list go out of scope, since its count is the number of active clients (ChatPerson instances) plus the ChatApp ownership of the array.

Additional members of ChatPerson include a string for creating the message to send (fSendString), the name of the client sending the message (fName), and a send function that prepares the string and adds it to the global chat list (see Figures 13 and 14).

Figure 13. chatperson.hpp

```
// chatperson.hpp
class ChatPerson: public EosObject
{
public:
ChatPerson();
ChatPerson(const ChatPerson& src);
virtual ~ChatPerson();
ChatPerson& operator =(const ChatPerson& src);

     // constructor which initializes the shared chat list
    ChatPerson(EosPrimitiveArray<EosString> *chatList);

// adds the user message to the chat list
    void send(void);
```

```
protected:
     // client message string for adding to the chat list
      EosString fSendString;
      // the client user name to be presented with the client's message in the
chat list
    EosString fName;
      // a "reference" to the shared chat list initialized within the ChatPerson
constructor
    EosPrimitiveArrayRef<EosString> fChatList;
};
```

Figure 14 chatperson.cpp

```
// chatperson.cpp
#include "stdeos.hpp"
#include <chatperson.hpp>

// performs the send which adds the client string to the chat list
void ChatPerson::send()
{
  // prepare the client string and add it to the chat list
  EosString theString("(" + fName + ") " + fSendString);
  fChatList->add(theString);

  // now clear the send string field on the client
  fSendString = "";
}

// the constructor that takes a reference to the global chat list
ChatPerson::ChatPerson(EosPrimitiveArray<EosString> *chatList) : EosObject(),
    fSendString(),
    fName(),
      // assign the shared chat list object with this client
      fChatList(chatList) {}

ChatPerson::ChatPerson() : EosObject(),
    fSendString(),
    fName(),
    fChatList() {}
ChatPerson::ChatPerson(const ChatPerson& src) : EosObject(src),
    fSendString(src.fSendString),
    fName(src.fName),
    fChatList(src.fChatList) {}
ChatPerson::~ChatPerson() {}
ChatPerson& ChatPerson::operator =(const ChatPerson& src)
{
    if (this != &src)
    {
        EosObject::operator=(src);
        fSendString = src.fSendString;
        fName = src.fName;
        fChatList = src.fChatList;
    }
    return *this;
}
```

Just a few lines of code are required to perform the basic chat application functionality. Notice that in this object connection-based approach the code that deals with application-specific logic (that of manipulating class

members, and so on) doesn't have to concern itself with network connections, sockets or RPC-based code, Java AWT client control settings, or communication. Those pieces are decoupled from the class structure itself and are automatically resolved by the connection technology enabled by the builder.

Although by default the details of the Java client applet (see Figure 15) are hidden by the builder's code generation, it is useful to look at the client code to see better what is happening there. Comments added to the generated code indicate the various pieces incorporated on the client in order to present and access remote C++ objects running on the server. It should be noted that some generated code has been omitted for brevity. A full example of the application can be inspected at http://www.viewsoft.com/examples.

Figure 15. Java Thin Client Chat Applet

```
// Java File Generated By ViewSoft's Builder
// Portions Copyright (c) 1996 MSJ
package chat;
import java.applet.*;
import java.awt.*;
import java.util.*;
import viewsoft.*;
public class ChatPerson extends EosEmbeddedView
{
  // this section of the generated code is used for client/server communication
  // and basic object to dialog connections...
  // socket port used for remote server side communication private EosPort fPort;
  // information used to connect view to C++ object on the server
  private EosMapperTable fIdTable;
  public ChatPerson() { super(); init(); }

  // data state change management hooks
  public EosMapperTableEntry getProbe(int id)
  { return (EosMapperTableEntry) fIdTable.elementAt(id); }
  public void removeProbe(int id) { fIdTable.removeProbe(id); }
  public void setMapperTable(EosMapperTable table) { fIdTable = table; }

  // port assignment
  public void setPort(EosPort port) { fPort = port; }

  // "resource" generation for the ChatPerson object dialogs designed in the
  // builder... responds to applet request to create an instance of a
  // ChatPerson dialog
  public void createView(String viewName, Container shell)
  {
    // since an object can have multiple views a comparison is made when the
    // create view function is called
    if (viewName.equals("Chat"))
    {
      int id;

      // create a geometry layout manager
      EosBoxContainer eosC1 = new EosBoxContainer();
      this.add(eosC1);

      // propagate communication variables
      eosC1.setPort(fPort);
      eosC1.setMapperTable(fIdTable);
      EosBoxContainer eosC2 = new EosBoxContainer();
      eosC1.add(eosC2);
      eosC2.setPort(fPort);
      eosC2.setMapperTable(fIdTable);

      // add the "Chat" label
      EosLabel eosC3 = new EosLabel();
```

```
eosC2.add(eosC3);
eosC3.setProperty("Caption", new EosString("Chat"));

// add the list box for the chat list
EosListBox eosC4 = new EosListBox();
eosC2.add(eosC4);

// probe ids...used to connect dialog elements to the object running on
// the server
id = fIdTable.attachProbe(eosC4, "fInvalidate");
eosC4.setUp(fPort, "fInvalidate", id);
id = fIdTable.attachProbe(eosC4, "select");
eosC4.setUp(fPort, "select", id);
id = fIdTable.attachProbe(eosC4, "insert");
eosC4.setUp(fPort, "insert", id);
id = fIdTable.attachProbe(eosC4, "fRemove");
eosC4.setUp(fPort, "fRemove", id);
id = fIdTable.attachProbe(eosC4, "fRemoveAll");
eosC4.setUp(fPort, "fRemoveAll", id);
id = fIdTable.attachProbe(eosC4, "set");
eosC4.setUp(fPort, "set", id);

// create a row/column layout manager
EosGridContainer eosC5 = new EosGridContainer();
eosC1.add(eosC5);

// propagate communication variables
eosC5.setPort(fPort);
eosC5.setMapperTable(fIdTable);

// "My Name" label
EosLabel eosC6 = new EosLabel();
eosC5.add(eosC6);
eosC6.setProperty("Caption", new EosString("My Name:"));

// horizontal layout manager
EosBoxContainer eosC7 = new EosBoxContainer();
eosC5.add(eosC7);
eosC7.setPort(fPort);
eosC7.setMapperTable(fIdTable);

// user name edit box
EosTextField eosC8 = new EosTextField();
eosC7.add(eosC8);
// probe ids
id = fIdTable.attachProbe(eosC8, "fText");
eosC8.setUp(fPort, "fText", id);

// send button
EosButton eosC9 = new EosButton();
eosC7.add(eosC9);
// probe ids
id = fIdTable.attachProbe(eosC9, "fPressed");
eosC9.setUp(fPort, "fPressed", id);
eosC9.setProperty("Caption", new EosString("Send"));

// "Message" label
```

```
      EosLabel eosC10 = new EosLabel();
      eosC5.add(eosC10);
      eosC10.setProperty("Caption", new EosString("Message:"));

      // edit box for the client message
      EosTextField eosC11 = new EosTextField();
      eosC5.add(eosC11);
      // probe ids
      id = fIdTable.attachProbe(eosC11, "fText");
      eosC11.setUp(fPort, "fText", id);
    }
  }
}
```

There are three main pieces to the Java applet that are generated to successfully deploy Internet thin-clients attached to C++ server executables. These are: socket communication, dialog resource description and creation, and remote C++ member attachment and connection protocols. These three components work in concert to create a Java applet that has minimal client-side resource requirements while still providing optimal communication performance over various Internet access speeds.

Figure 16 demonstrates the client view of the chat application running in Internet Explorer.

{ewc mvimg, mvimage,!illust.bmp}
Figure 16. The Client View of the Chat Application Running in Internet Explorer

To summarize, the thin client three-tier model builds on your expertise as a Visual C++ developer for creating powerful applications for the Internet. The model lets you effectively use Java, giving your applications the dynamic cross-platform deployment mechanism needed for the Internet. It also provides the client side of your applications with just enough application logic to optimally process and communicate user interaction across the Internet to the server. And, best of all, you don't have to be a Java guru to build real Internet applications that solve real problems with Java.

The excitement surrounding the Internet is here to stay. The challenge to you is to apply the litmus test as you evaluate the profusion of frameworks and tools available in the months ahead. Then, while everyone else is trying to figure out which way they're going, you can get some real work done.

To demonstrate how easy it can be to get started with WinSock programming, I've developed Flipper: The MSJ Finger Program. I chose to write a finger program because it's one of the simplest things you can do on the Internet. It doesn't require you to maintain a connection to a remote machine the way an FTP (file transfer protocol) session would. All you have to do is find a remote site's unique numeric IP address, send a string containing a user's name to the finger port at this IP address, and wait for return data. When your machine gets data back from that IP address, your program is notified that it has received data, and you retrieve it and display it (see **Figure 1**).

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. Fingering a Remote User Through WinSock

Perhaps I should define just what a finger program is. It's a simple query sent to a remote node to retrieve information about a user or users on the system. Some Internet services, like file browsing and transfer (FTP) and remote terminal access (Telnet) need a full-blown connection, with signon procedures and a persistent connection. Finger, however, doesn't need anything that fancy. When you send a CR/LF-terminated user name as a message to a site's designated finger port, it sends back a block of text about that user. This data is defined by the individual machine; it can contain connection information, real-life names, the amount of mail a user has waiting, or even display user-defined information (commonly known as a .plan file). After this information is returned, you're under no further obligation. (Just write CANCEL on the bill and you owe nothing!)

So let's go over the steps needed to implement a finger program. First, you have to connect to the TCP/IP stack and receive a socket handle to use. You then have to retrieve the address of the remote site, connect to it, send it the username you're interested in, wait for it to return finger information, and disconnect.

To start with, I created an MSJSock control with the Microsoft Developer Studio (AKA Visual C++ 4.0). To get the finger command working correctly, the control needs five methods: CreateSocket, gethostbyname, getservbyname, AsyncSelect, and Connect. Technically, some of these WinSock calls can be made from Visual Basic itself, but putting them in an MSJSock OCX makes some operations easier. For instance, if you don't know which standard port to use for a particular service, you can use the getservbyname socket API function. If you call

```
getservbyname("finger", "tcp")
```

you're returned information about the TCP finger protocol, but it's returned as a structure with nasty contents such as pointers to pointers. It's easier to deal with this information in C/C++, because Visual Basic doesn't handle pointer values very well. In Windows 3.1, you could potentially have called hmemcpy, a memory access API function exported by KERNEL.DLL. Sadly, this function is no longer exported by KERNEL32.DLL, so you either have to export your own equivalent method or create a call that returns a particular member of a structure. I've chosen the second path, which is less extensible but leads to easier Visual Basic code. Similarly, when you need to call an asynchronous WinSock function, the OLE control will provide the message trap for you, and will fire an event into the Visual Basic project when network data arrives. Clearly, the OCX and Visual Basic project need each other to function correctly, although the control will not be bound to one particular hunk of Visual Basic code.

A client only needs two pieces of information to perform a finger — a user ID and a site name. Therefore, the Flipper Visual Basic program can be completed with a fairly bare bones implementation. I've added two edit boxes (one for each piece of info), a text box to display results, a button that starts a finger request, and an MSJSock control. When the edit boxes are filled with a username and remote site name, the button is enabled. When it's clicked, the program swings into action. It calls the MSJSock control's CreateSocket method, which in turn calls the WinSock socket function. (MSJSock code may be found in **Figure 2**; the Flipper project is shown in **Figure 3**.) socket returns a unique socket ID immediately, so you have a handle to pass data to perform the finger task.

Figure 2. MSJSock

MSJSOCK.CPP (excerpts)

```
// MSJSock.cpp : Implementation of CMSJSockApp and DLL registration.

#include "stdafx.h"
#include "MSJSock.h"

////////////////////////////////////////////////////////
// CMSJSockApp::InitInstance - DLL initialization

BOOL CMSJSockApp::InitInstance()
{
    BOOL bInit = COleControlModule::InitInstance();

    // Normally we'd call WSAStartup, but AfxSocketInit
    // wraps the call for us and takes care of some of the
    // version checking automatically.

    if (bInit)
    {
        if (!AfxSocketInit(&m_wsa))
        {
            AfxMessageBox(IDP_SOCKETS_INIT_FAILED);
            return FALSE;
        }
    }

    return bInit;
}


////////////////////////////////////////////////////////
// CMSJSockApp::ExitInstance - DLL termination

int CMSJSockApp::ExitInstance()
{
    AfxSocketTerm();
    return COleControlModule::ExitInstance();
}

MSJSOCK.H (excerpts)
////////////////////////////////////////////////////////
// CMSJSockApp : See MSJSock.cpp for implementation.

class CMSJSockApp : public COleControlModule
{
public:
    BOOL InitInstance();
```

```
    int ExitInstance();

    LPWSADATA Sock() { return &m_wsa; };

protected:
    WSADATA m_wsa; // Windows socket info - filled on startup
};

MSJSOCKCtTL.CPP (excerpts)
// MSJSockCtl.cpp : Implementation of the CMSJSockCtrl OLE control class.

                                        .
                                        .
                                        .

extern CMSJSockApp theApp;
LPBYTE pm_szBuf;


/////////////////////////////////////////////////////////////////////////
// Message map

BEGIN_MESSAGE_MAP(CMSJSockCtrl, COleControl)
    //{{AFX_MSG_MAP(CMSJSockCtrl)
    // NOTE - ClassWizard will add and remove message map entries
    //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG_MAP
    ON_MESSAGE(WSCB_GETHOST, OnGetHostCB)
    ON_MESSAGE(WSCB_ASELECT, OnASelectCB)
    ON_OLEVERB(AFX_IDS_VERB_EDIT, OnEdit)
    ON_OLEVERB(AFX_IDS_VERB_PROPERTIES, OnProperties)
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////
// Dispatch map

BEGIN_DISPATCH_MAP(CMSJSockCtrl, COleControl)
    //{{AFX_DISPATCH_MAP(CMSJSockCtrl)
    DISP_PROPERTY_EX(CMSJSockCtrl, "Version", GetVersion, SetVersion, VT_I2)

DISP_PROPERTY_EX(CMSJSockCtrl, "SystemStatus", GetSystemStatus, SetSystemStatus,
                 VT_BSTR)

DISP_PROPERTY_EX(CMSJSockCtrl, "MaxSockets", GetMaxSockets, SetMaxSockets, VT_I2)
    DISP_PROPERTY_EX(CMSJSockCtrl, "HiVersion", GetHiVersion, SetHiVersion, VT_I2)
    DISP_PROPERTY_EX(CMSJSockCtrl, "Description", GetDescription, SetDescription,
                     VT_BSTR)
    DISP_FUNCTION(CMSJSockCtrl, "CreateSocket", CreateSocket, VT_I4, VTS_NONE)

DISP_FUNCTION(CMSJSockCtrl, "gethostbyname", gethostbyname, VT_HANDLE, VTS_BSTR)
    DISP_FUNCTION(CMSJSockCtrl, "getservbyname", getservbyname, VT_I2, VTS_BSTR)
    DISP_FUNCTION(CMSJSockCtrl, "AsyncSelect", AsyncSelect, VT_I4, VTS_I4 VTS_I4)
    DISP_FUNCTION(CMSJSockCtrl, "connect", connect, VT_HANDLE, VTS_I4 VTS_I2
                  VTS_HANDLE)
    //}}AFX_DISPATCH_MAP
    DISP_FUNCTION_ID(CMSJSockCtrl, "AboutBox", DISPID_ABOUTBOX, AboutBox,
                     VT_EMPTY, VTS_NONE)
```

```
END_DISPATCH_MAP()


////////////////////////////////////////////////////////////////////////
// Event map

BEGIN_EVENT_MAP(CMSJSockCtrl, COleControl)
    //{{AFX_EVENT_MAP(CMSJSockCtrl)
    EVENT_CUSTOM("Connect", FireConnect, VTS_HANDLE)
    EVENT_CUSTOM("RecvData", FireRecvData, VTS_HANDLE  VTS_BSTR)
    EVENT_CUSTOM("SockError", FireSockError, VTS_HANDLE  VTS_I2)
    EVENT_CUSTOM("GotHost", FireGotHost, VTS_HANDLE)
    //}}AFX_EVENT_MAP
END_EVENT_MAP()

                                        .
                                        .
                                        .


////////////////////////////////////////////////////////////////////////
// Interface IDs

const IID BASED_CODE IID_DMSJSock =
   { 0x9e638961, 0x2c21, 0x11cf, { 0xa0, 0x18, 0x44, 0x45, 0x53, 0x54, 0, 0 } };
const IID BASED_CODE IID_DMSJSockEvents =
   { 0x9e638962, 0x2c21, 0x11cf, { 0xa0, 0x18, 0x44, 0x45, 0x53, 0x54, 0, 0 } };


////////////////////////////////////////////////////////////////////////
// Control type information

// Important: make sure OLEMISC_INVISIBLEATRUNTIME is not included.
//    This would cause the control not to have an hWnd at runtime;
//    we want one, but we want it hidden.

static const DWORD BASED_CODE _dwMSJSockOleMisc =
    OLEMISC_ACTIVATEWHENVISIBLE |
    OLEMISC_SETCLIENTSITEFIRST |
    OLEMISC_INSIDEOUT |
    OLEMISC_CANTLINKINSIDE |
    OLEMISC_RECOMPOSEONRESIZE;

IMPLEMENT_OLECTLTYPE(CMSJSockCtrl, IDS_MSJSOCK, _dwMSJSockOleMisc)

                                        .
                                        .
                                        .

////////////////////////////////////////////////////////////////////////
// CMSJSockCtrl::OnDraw - Drawing function

void CMSJSockCtrl::OnDraw(
            CDC* pdc, const CRect& rcBounds, const CRect& rcInvalid)
{
    // Hide the control at runtime
```

```
    if (m_hWnd != NULL)
        ShowWindow(SW_HIDE);
    else
    {
        // Draw something at design-time
        pdc->FillRect(rcBounds,
CBrush::FromHandle((HBRUSH)GetStockObject(WHITE_BRUSH)));
        pdc->Ellipse(rcBounds);
    }
}

                                        .
                                        .
                                        .

/////////////////////////////////////////////////////////////////////////
// CMSJSockCtrl message handlers

// Get the current Windows Sockets version from the info returned at startup

short CMSJSockCtrl::GetVersion()
{
    return theApp.Sock()->wVersion;
}

void CMSJSockCtrl::SetVersion(short nNewValue)
{
    // Read/only property, so do nothing.
}

// Get the current system status from the info returned at startup

BSTR CMSJSockCtrl::GetSystemStatus()
{
    CString s;
    s = theApp.Sock()->szSystemStatus;
    return s.AllocSysString();
}

void CMSJSockCtrl::SetSystemStatus(LPCTSTR lpszNewValue)
{
    // Read/only property, so do nothing.
}

// Get the maximum sockets allowed on this control

short CMSJSockCtrl::GetMaxSockets()
{
    return theApp.Sock()->iMaxSockets;
}

void CMSJSockCtrl::SetMaxSockets(short nNewValue)
{
    // Read/only property, so do nothing.
}

// Get the highest allowed version from the info returned at startup
```

```
short CMSJSockCtrl::GetHiVersion()
{
    return theApp.Sock()->wHighVersion;
}

void CMSJSockCtrl::SetHiVersion(short nNewValue)
{
    // Read/only property, so do nothing.
}

// Get the Windows Sockets description from the startup info

BSTR CMSJSockCtrl::GetDescription()
{
    CString s;

    s = theApp.Sock()->szDescription;
    return s.AllocSysString();
}

void CMSJSockCtrl::SetDescription(LPCTSTR lpszNewValue)
{
    // Read/only property, so do nothing.
}

// Create a socket - wrap the socket() API call

long CMSJSockCtrl::CreateSocket()
{
    SOCKET hSock;

    hSock = ::socket(AF_INET, SOCK_STREAM, 0);

    if (hSock <= 0)
        return -1;
    else
        return hSock;
}

// Convert a host string to its IP address

OLE_HANDLE CMSJSockCtrl::gethostbyname(LPCTSTR host)
{
    pm_szBuf = (LPBYTE) GlobalAllocPtr(GHND, 2048);

    HANDLE hTask = WSAAsyncGetHostByName(m_hWnd, WSCB_GETHOST, host,
                                         (char FAR *) pm_szBuf, 2048);

    return (OLE_HANDLE) hTask;
}

// Wait for the GetHostByName to return info. This maps WSCB_GETHOST
// into the OnGetHostCB message handler.

afx_msg LONG CMSJSockCtrl::OnGetHostCB(UINT wParam, LONG lParam)
{
    HANDLE hTask = (HANDLE) wParam;
```

```
    UINT   wserr = HIWORD(lParam);
    UINT   uCode = LOWORD(lParam);

    LPVOID lphe;

    u_long addr;
    hostent he;
    sockaddr_in ca;

    // If the call didn't return an error, call GotHost in the
    // container program.

    if (wserr = = WSANOERROR)
    {
        lphe = (LPVOID) GlobalAllocPtr(GHND, sizeof(hostent));

        memcpy(lphe, (LPVOID) pm_szBuf, sizeof(hostent));
        FireGotHost((OLE_HANDLE) lphe);

        return TRUE;
    }
    else // call SockError in the container
        FireSockError((OLE_HANDLE) hTask, wserr);

    return TRUE;
}

// Handle all messages requested by a WSAAsyncSelect call

afx_msg LONG CMSJSockCtrl::OnASelectCB(UINT wParam, LONG lParam)
{
    HANDLE hTask = (HANDLE) wParam;
    UINT   wserr = HIWORD(lParam);
    UINT   uCode = LOWORD(lParam);

    u_long addr;
    hostent he;
    sockaddr_in ca;

    switch (uCode)
    {
        // If a connect message comes back, fire the Connect event
        case FD_CONNECT:
              if (wserr != WSANOERROR)
                FireSockError((OLE_HANDLE) hTask, wserr);
            else
            {
                FireConnect((OLE_HANDLE) hTask);
            }
            break;

        // Not yet handling a close message
        case FD_CLOSE:
            break;

        // When data is incoming, call the RecvData event
        case FD_READ:
```

```
        {
            FireRecvData((OLE_HANDLE) hTask, NULL);
            break;
        }

        // Not yet handling a ready to write message
        case FD_WRITE:
            break;

        // Not yet handling an out of band data message
        case FD_OOB:
            break;

        // Not yet handling an accept message
        case FD_ACCEPT:
            break;

        default:
            break;
    }

    return TRUE;
}

// Wrap the getservbyname function - returning a port ID when
// given a service like "finger." This only handles tcp now, not
// udp.

short CMSJSockCtrl::getservbyname(LPCTSTR service)
{
    short port;
    LPSERVENT servent;

    servent = ::getservbyname(service, "tcp");
    port = servent->s_port;

    // Convert the port from network byte order to host
    // machine byte order and return.
    return ntohs(port);
}

// Wrap the WSAAsyncSelect call for the container's sake. This
// automatically routes incoming messages to this control's
// window.

long CMSJSockCtrl::AsyncSelect(long socket, long msg)
{
    return WSAAsyncSelect(socket, m_hWnd, WSCB_ASELECT, msg);
}

// Wrap the functions needed to perform a connect to a port/host.

OLE_HANDLE CMSJSockCtrl::connect(long socket, short port, OLE_HANDLE phostent)
{
    hostent he;
    u_long addr;
    sockaddr_in ca;
```

```
    // Copy the hostent data handle from the container to a real
    // hostent structure.

    memcpy((LPVOID) &he, (LPVOID) phostent, sizeof(hostent));

    // Get the IP address from the hostent.

    addr = (*(u_long FAR *) he.h_addr_list[0]);

    // Fill in the sockaddr_structure so we can connect to
    // the specific port and IP address
    ca.sin_family      = AF_INET;
    ca.sin_port        = htons(port); // Convert from our format to net format
    ca.sin_addr.s_addr = addr;

    // Make the control watch for connections
    WSAAsyncSelect(socket, m_hWnd, WSCB_ASELECT, FD_CONNECT);

    // Call the API connect() function, returning immediately.
    // The container will get a Connect event when the
    // connection is complete.

    return ::connect(socket, (LPSOCKADDR) &ca, sizeof(sockaddr_in));
}
```

Figure 3. FLIPPER

FLIPPER.FRM (procedures)

```
'-----------------------------------------------------
' Flipper.FRM
'   This is the form definition for Flipper: The MSJ
'   Finger program.


'-----------------------------------------------------
' When the "Finger" button is clicked, Flipper should
' perform three tasks: create a socket, get the finger
' service port, and get the remote host's IP address
' given its name.

Private Sub btnFinger_Click()
    Dim rc
    sock = MSJSockCtrl.CreateSocket
    fingport = MSJSockCtrl.getservbyname("finger")
    rc = MSJSockCtrl.gethostbyname(edtSite.Text)
End Sub

'-----------------------------------------------------
' When a user types in the site or userid edit boxes,
' the "Finger" button is enabled or disabled depending
' on whether both boxes have data.

Private Sub edtSite_Change()
    If Len(edtUser) = 0 Or Len(edtSite) = 0 Then
        btnFinger.Enabled = False
    Else
```

```
        btnFinger.Enabled = True
    End If
End Sub

Private Sub edtUser_Change()
    If Len(edtUser) = 0 Or Len(edtSite) = 0 Then
        btnFinger.Enabled = False
    Else
        btnFinger.Enabled = True
    End If
End Sub

'-----------------------------------------------------
' When the form is loaded, disable the Finger
' button until both userid and site are filled in.

Private Sub Form_Load()
    btnFinger.Enabled = False
End Sub


'-----------------------------------------------------
' MSJSockCtrl_connect is an event called by the MSJSock
' OLE control when a successful connection has been made.
' If you want, you can track a particular task by its handle
' (given here as hTask)

Private Sub MSJSockCtrl_connect(ByVal hTask As Long)
    Dim rcl As Long, rc As Integer

    ' Tell MSJSock cntrl to watch for "ready to read," "ready for more writing,"
    ' and "connection closed" events.
    rcl = MSJSockCtrl.AsyncSelect(sock, FD_READ + FD_WRITE + FD_CLOSE)

    ' send() is a direct WINSOCK API call. It sends the indicated
    ' text to the socket specified. In this case, we're just sending
    ' a username to the finger port...
    rc = send(sock, edtUser.Text, Len(edtUser.Text), 0)

    ' ...and appending a CR-LF pair.
    rc = send(sock, Chr$(13) & Chr$(10), 2, 0)
End Sub

'-----------------------------------------------------
' When we called gethostbyname when the Finger button is
' clicked, it had to go off and find the host's IP address. When
' the OLE control gets that data back, it returns it by calling
' the GotHost event.

Private Sub MSJSockCtrl_GotHost(ByVal hostent As Long)
    ' Now that we have a pointer to the host's address (in hostent),
    ' we can take the socket we created earlier and connect it to
    ' the appropriate host and its finger port.
    Call MSJSockCtrl.Connect(sock, fingport, hostent)
End Sub

'-----------------------------------------------------
```

```
' Whenever our TCP/IP stack receives data from the network,
' it notifies the MSJSock control. In turn, if the control is
' expecting some data, it informs the VB container program
' that data needs to be read by calling the RecvData event.

Private Sub MSJSockCtrl_RecvData(ByVal hTask As Long, ByVal Data As String)
    Dim msg As String
    Dim rc As String

    ' Initialize the variable-length string with 1024 blanks - enough room to
    ' read in data.
    msg = Space(1024)

    ' recv() is part of WINSOCK.DLL. Since we've just been notified that
    ' there's data to receive, Flipper calls it to receive the data.
    rc = recv(sock, msg, 1024, 0)

    ' Trim the network input and display it in the output text box.
    lblOutput = Trim(msg)
End Sub

'-----------------------------------------------------
' If the MSJSock control returns a WinSock error, filter it through our
' WSERROR.BAS and display it for the user.

Private Sub MSJSockCtrl_SockError(ByVal hTask As Long, ByVal ENum As Integer)
    MsgBox "WinSock error " & ENum & "(" & WSErrStr(ENum) & ")", vbOKOnly,_
    "WinSock Error"
End Function

WINSOCK.BAS
'-------------------------------------------------------
' WINSOCK.BAS
'   This file contains Declares for some of the more
'   popular Windows Socket functions.

Global sock As Long ' Our own personal socket!
Global fingport As Integer

' Constants for use with the WSAAsyncSelect call

Global Const FD_READ = &H1
Global Const FD_WRITE = &H2
Global Const FD_OOB = &H4
Global Const FD_ACCEPT = &H8
Global Const FD_CONNECT = &H10
Global Const FD_CLOSE = &H20


' These are some of the standard Windows Socket calls we need in
' Flipper, defined for both 16- and 32-bit versions of the DLL.

#If Win16 Then
Declare Function oBind Lib "winsock.dll" Alias "bind" (ByVal s As Integer,_
Addr As sockaddr_in, ByVal namelen As Integer) As Integer
Declare Function htonl Lib "winsock.dll" (ByVal a As Long) As Long
Declare Function inet_addr Lib "winsock.dll" (ByVal s As String) As Long
Declare Function inet_ntoa Lib "winsock.dll" (ByVal inn As Long) As Long
```

```
Declare Function ntohl Lib "winsock.dll" (ByVal a As Long) As Long
Declare Function oSocket Lib "winsock.dll" Alias "socket" (ByVal af As_
Integer, ByVal typesock As Integer, ByVal protocol As Integer) As Integer
Declare Function htons Lib "winsock.dll" (ByVal a As Integer) As Integer
Declare Function ntohs Lib "winsock.dll" (ByVal a As Integer) As Integer
Declare Function oConnect Lib "winsock.dll" Alias "connect" (ByVal sock As_
Integer, sockstruct As sockaddr_in, ByVal structlen As Integer) As Integer
Declare Function send Lib "winsock.dll" (ByVal sock As Integer, ByVal msg As_
String, ByVal msglen As Integer, ByVal flag As Integer) As Integer
Declare Function recv Lib "winsock.dll" (ByVal sock As Integer, ByVal msg As_
String, ByVal msglen As Integer, ByVal flag As Integer) As Integer
Declare Function getsockname Lib "winsock.dll" (ByVal s As Integer, Addr As_
Any, ByVal namelen As Integer) As Integer
Declare Function getservbyname Lib "winsock.dll" (ByVal nm As String, ByVal_ proto
As String) As Long

Declare Function WSAStartup Lib "winsock.dll" (ByVal a As Integer, b As_
WSAdata_type) As Integer
Declare Function WSACleanup Lib "winsock.dll" () As Integer
Declare Function WSAGetLastError Lib "winsock.dll" () As Integer

Declare Function lstrcpyn Lib "kernel" (ByVal lpszString1 As Any, ByVal_
lpszString2 As Long, ByVal cChars As Integer) As String
Declare Sub hmemcpy Lib "kernel" (hpvDest As Any, hpvSource As Any, ByVal_ cbCopy
As Long)

#Else
Declare Function htonl Lib "wsock32.dll" (ByVal a As Long) As Long
Declare Function inet_addr Lib "wsock32.dll" (ByVal s As String) As Long
Declare Function inet_ntoa Lib "wsock32.dll" (ByVal inn As Long) As Long
Declare Function ntohl Lib "wsock32.dll" (ByVal a As Long) As Long
Declare Function oSocket Lib "wsock32.dll" Alias "socket" (ByVal af As_
Integer, ByVal typesock As Integer, ByVal protocol As Integer) As Integer
Declare Function htons Lib "wsock32.dll" (ByVal a As Integer) As Integer
Declare Function ntohs Lib "wsock32.dll" (ByVal a As Integer) As Integer
Declare Function send Lib "wsock32.dll" (ByVal sock As Integer, ByVal msg As_
String, ByVal msglen As Integer, ByVal flag As Integer) As Integer
Declare Function recv Lib "wsock32.dll" (ByVal sock As Integer, ByVal msg As_
String, ByVal msglen As Integer, ByVal flag As Integer) As Integer

Declare Function ogetservbyname Lib "wsock32.dll" Alias "getservbyname"_
(ByVal nm As String, ByVal proto As String) As Long
Declare Function gethostname Lib "wsock32.dll" (ByVal s As String, ByVal l As_
Integer) As Integer
Declare Function gethostbyname Lib "wsock32.dll" (ByVal s As String) As Long

Declare Function WSACleanup Lib "wsock32.dll" () As Integer
Declare Function WSAGetLastError Lib "wsock32.dll" () As Integer
Declare Function WSAAsyncGetHostByName Lib "wsock32.dll" (ByVal hWnd As_
Integer, ByVal wMsg As Integer, ByVal name As String, ByVal buf As String,_
ByVal buflen As Integer) As Integer

Declare Function lstrcpyn Lib "kernel32" Alias "lstrcpynA" (ByVal lpString1_
As Any, ByVal lpString2 As Long, ByVal iMaxLength As Long) As Long
Declare Sub hmemcpy Lib "kernel32" (hpvDest As Any, hpvSource As Any, ByVal_
cbCopy As Long)

#End If
```

When you write a WinSock-compliant program, the first thing your app needs to do is initialize socket services for its instance with a WSAStartup call. WSAStartup fills in a structure with some light information like the current WinSock version and a description string such as “Running Under Windows 95.” Since I’m using my MSJSock OCX to make some of the structure-passing calls easier, I’ve put the WSAStartup call in MSJSock’s InitInstance function.

Next, the Visual Basic program calls the control’s getservbyname method to get the correct port ID for the finger command. As I mentioned earlier, a site can support more than one function — for instance, one server can provide both finger and FTP services. This is done through dedicated port numbers. It is understood that when a user sends a message to the finger port (usually number 79), they’re sending a finger command. However, on the off chance that a server has a nonstandard implementation, or you’re using a specialized service, or you just want to make sure your local server supports a particular interface, WinSock provides getservbyname so you can be confident there’s no funny business going on.

Now you have to look up the IP address for the site you’re fingering. To keep track of all sites in a uniform manner, each is assigned a long integer address when it is founded. Service providers (such as MSN) know how to translate a string address such as “ftp.microsoft.com” into the matching long integer. Sometimes this number is broken up into four bytes, so that ftp.microsoft.com is reachable at 32008646, but is commonly known as 198.105.232.1 ($198 + 105 \times 256 + 232 \times 256^2 + 1 \times 256^3$). You’ll be pleased to know, by the way, that when the current stock of long integers is exhausted in a few years, IP addresses will become longer, breaking everyone’s software even worse than the change-over to the 21st century that is going to nuke all those two-digit year fields so popular on mainframe programs.

Retrieving a host by name is an asynchronous function. When you call the control’s gethostbyname method, it sends the request off and returns immediately. MSJSock has a separate message — handling function that will be called by WinSock when the information is ready. This function (OnGetHostCB) calls the GotHost event in Visual Basic. The Visual Basic program is free to do whatever it pleases until MSJSock_GotHost is invoked. When MSJSock_GotHost is called, the OLE control passes Visual Basic a long integer called hostent, which is really a pointer to a structure describing the host address. You don’t need to dereference this; you just need to pass it back when you connect to the host site.

To get an actual connection, you need three shards of information: the socket handle retrieved earlier, the port ID for fingering, and the hostent structure containing the site’s IP address. The connection to the remote host can be initiated by calling the Connect method of the MSJSock control. MSJSock.Connect does two things. First, it calls WSAAsyncSelect. WinSock uses this function to determine where to send asynchronous notification messages for a particular socket. The caller chooses what sort of data to watch for. Here, you pass the FD_CONNECT flag to WSAAsyncSelect; this tells WinSock to let the control know when a connect action has completed. The control can then check the error code — if it’s 0, it fires the MSJSock1_Connect event in Visual Basic. Otherwise, it calls MSJSock1_SockError to indicate an error occurred. I’ve set up a Select Case block in WINSOCK.BAS that will match known WinSock error codes with their names, so the program can pop up a MsgBox.

You’re more than halfway there now: determining addresses and connections is the hardest part of WinSock programming. When MSJSock_Connect is triggered inside Flipper, it means you’ve connected correctly to the remote site’s finger port. All that’s left to finger a node is to send it a user name terminated with a CR-LF pair. In the control, I’ve exposed an AsyncSelect method that wraps the WSAAsyncSelect function. This time you want to trap a message any time WinSock tells you there’s data from the remote site to be read, so pass it FD_READ. You can now call the WinSock send function directly from Visual Basic, specifying the socket ID and string. Since the earlier Connect call linked the specific remote site and port to the socket received, you don’t have to tell WinSock where to route the data again.

The send call isn’t really asynchronous, because it doesn’t guarantee that data will be coming back. However, the remote host usually does something in response — in the case of finger, it sends back user information and status. When the MSJSock control captures an FD_READ notification from WinSock, it generates a RecvData event in Flipper. This only tells the Visual Basic program that data is available for retrieval, so it’s that program’s job to call the WinSock recv function in response. recv takes three parameters: the socket ID, an empty string where the data will be stuck, and the maximum length of this string. Instead of declaring a fixed-length string in Visual Basic, I’ve declared a variable length string, then initialized it with blanks:

```
strg$ = Space(1024)
```

After this string is passed to recv, it can be trimmed (with the Trim function), then displayed in the text box I added to the program for just such a purpose. If all goes well, the Flipper program is complete (see **Figure 4**).

Even if something goes wrong, it will display an error code.

{ewc mvimg, mvimage,!illust.bmp}
Figure 4. Flipper

OK, so calling finger is sort of cinchy compared with other Internet protocols. It doesn't maintain a connection and doesn't make you learn a batch of commands to work properly. Unfortunately, describing each and every available service would make a book or two in itself — or a whole shelf of books, if you check out your local bookstore. There's no reason to let this scare you away from the job, however. There's a big need out there for commercial-quality 32-bit news readers, Telnet programs, and just about anything you might be interested in developing.

The various service protocols are described in documents called RFCs (Request For Comments) and STDs (standards). The first thing you should do is determine what type of program you're writing, then grab the documentation on it so you know what commands a server supports (see **Figure 5**). There's nearly 2000 Internet, TCP/IP, and sockets RFCs currently available, stretching all the way back to 1969. There are a few sources for RFCs. If you have Internet software, you can ftp anonymously to ds.internic.net, then retrieve files from the /rfc directory. An excellent book, *Developing for the Internet with Winsock* by Dave Roberts (Coriolis Group Books, 1995), comes with a CD-ROM with every piece of Internet and sockets documentation you'll ever find. **Figure 5** shows the most common Internet services and the matching RFCs.

Figure 5. Common Internet Services

Internet Service	Info location
Mail (send)	STD 010, SMTP-Simple Mail Transfer Protocol
Mail (receive)	RFC 1460, POP3-Post Office Protocol
Usenet	RFC 0977, Network News Transmission Protocol
World Wide Web (HTML)	RFC 1866, "Hypertext Markup Language - 2.0"
WWW (HTTP)	ftp://info.cern.ch/pub/www/doc/http.txt
Gopher	RFC 1436, "The Internet Gopher Protocol (a distributed document search and retrieval protocol)"
Telnet	RFC 0854, "Telnet Protocol specification"
Chat	RFC 1459, "Internet Relay Chat Protocol"
Finger	RFC 1288
Internet Phone	RFC 1789, "INETPhone: Telephone Services and Servers on Internet"
FTP	RFC 1350, "The TFTP Protocol (revision 2)" STD 009

To find the name of your Web server, run the Network Control Panel applet on your Web server. The computer name is listed on the **Identification** tab.

Windows Sockets programming is currently the only pathway to every protocol and function provided on the Internet. However, up till now it's been seen as just beyond the reach of Visual Basic programmers. I've provided two components to change this — the MSJSock OLE control and the MSJ Simple Sockets Visual Basic application. With surprisingly little code, you can write a real Internet-enabled program in Visual Basic. If you get something on the market, all I ask is that you send me a couple of shares of your Internet company stock.

To assist you in debugging your socket-based program, I've developed a small program called Simple Sockets. It's an interactive dialog that lets you send commands to a server and see what gets returned from it. The entire transaction is displayed in a Windows 95 RichText edit box, tagged by color so you can easily see what's up.

As an example, I've downloaded RFC 0977 (the Usenet NNTP news protocol) and now I want to see just how it works. As you may or may not know, Usenet is the name for a network of servers that trade user-posted articles on every topic ever imagined (about 16,000 in all at press time). Using a newsreader, you can browse articles by title or number, post followups, or save and print especially useful tidbits. It's a distributed system with no single home, so you'll be sure to see many fruitless efforts by bureaucrats to legislate its content over the next few years.

But anyway. I need two things to connect: the name of my news server (nntp.ix.netcom.com) and the NNTP port (119 is standard; this is given in the RFC). I enter the two values and click connect, and Windows 95 either performs an auto-login with the Dial-Up Adapter for me or connects me to a running instance of WinSock. Non-text-returning actions like this are displayed as informational messages in blue. When the program indicates "Connected to host," I can start sending my commands, which show up in green. Whenever my program receives something from the host, it's appended to the RichText edit in red, and should a WinSock error occur (a bad server name, for instance), the message will show up in black (see **Figure 6**).

{ewc mvimg, mvimage,!illust.bmp}
Figure 6. MSJ Simple Sockets

On my server, I have to sign on with the AUTHINFO command, providing my username and password. I enter it in the edit field provided, then click the Send button or hit enter. After a couple of seconds, I receive back an informational message from nntp.ix.netcom.com: "501 user Name|pass Password." Looking up a message code 501 in the RFC, I'm told that my command had a syntax error. I'm not upset, because it seems to work anyway.

Without going into too much detail, an NNTP server will provide a list of all available newsgroups with the LIST command. Be sparing with it, though — it returns the names of all 16,000 groups, plus or minus. Once you get this list downloaded once, you can use the NEWGROUPS command to see all additions from a certain date.

Figure 6 shows some other NNTP commands in action. I've decided to check out the action on alt.fan.surak. (And I don't need any editorial comment from you either.) The GROUP command returns an informational code (211, which means it's a reply to a GROUP command, as documented in the RFC). It also tells you the number of articles available, the first and last articles available, and the group name. Instead of doing this for 16,000 groups every time a user starts a program, news readers usually have a local "subscribe" option, which really only makes an internal list of which groups it should check for content automatically.

After selecting a group with GROUP, it is implicitly assumed that you're operating on it with further commands. For instance, I've issued an ARTICLE command to receive a particular numbered article (along with a couple of bad commands just to show what they might output). Article 2615 is returned to my program, headers, text and all. I'm now just a little bit more caught up with the world of Surak fandom. I know, it's too good to be true. Anyway, MSJ Simple Sockets source code is shown in **Figure 7**.

Figure 7. Simple Sockets.FRM

VERSION 4.00

```
Begin VB.Form frmSimple
   Caption         =   "MSJ Simple Sockets"
   ClientHeight    =   6705
   ClientLeft      =   1215
   ClientTop       =   1545
   ClientWidth     =   8445
   Height          =   7125
   Left            =   1155
   LinkTopic       =   "Form1"
   ScaleHeight     =   6705
   ScaleWidth      =   8445
   Top             =   1185
   Width           =   8565
   Begin VB.CommandButton btnSend
      Caption         =   "&Send"
```

```
   Height          =   375
   Left            =   7440
   TabIndex        =   8
   Top             =   1080
   Width           =   855
End
Begin VB.TextBox edtOutput
   Height          =   285
   Left            =   120
   TabIndex        =   7
   Top             =   1200
   Width           =   7215
End
Begin VB.TextBox edtPort
   Height          =   285
   Left            =   840
   TabIndex        =   4
   Text            =   "119"
   Top             =   480
   Width           =   2895
End
Begin VB.TextBox edtSite
   Height          =   285
   Left            =   840
   TabIndex        =   1
   Text            =   "nntp.ix.netcom.com"
   Top             =   120
   Width           =   2895
End
Begin VB.CommandButton btnConnect
   Caption         =   "Connect"
   Height          =   375
   Left            =   3840
   TabIndex        =   0
   Top             =   120
   Width           =   1215
End
Begin VB.Label Label5
   BackStyle       =   0  'Transparent
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Outbound"
   ForeColor       =   &H00008000&
   Height          =   255
   Left            =   5520
   TabIndex        =   12
   Top             =   120
   Width           =   1215
End
Begin VB.Label Label4
   BackStyle       =   0  'Transparent
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Inbound"
   ForeColor       =   &H000000FF&
   Height          =   255
   Left            =   6840
   TabIndex        =   11
   Top             =   480
```

```
   Width           =   1455
End
Begin VB.Label Label3
   BackStyle       =   0  'Transparent
   BorderStyle     =   1  'Fixed Single
   Caption         =   "WinSock Errors"
   Height          =   255
   Left            =   5520
   TabIndex        =   10
   Top             =   480
   Width           =   1215
End
Begin VB.Label Label2
   BackStyle       =   0  'Transparent
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Info"
   ForeColor       =   &H00FF0000&
   Height          =   255
   Left            =   6840
   TabIndex        =   9
   Top             =   120
   Width           =   1455
End
Begin RichtextLib.RichTextBox redInOut
   Height          =   5055
   Left            =   120
   TabIndex        =   6
   Top             =   1560
   Width           =   8175
   _Version        =   65536
   _ExtentX        =   14420
   _ExtentY        =   8916
   _StockProps     =   69
   BackColor       =   -2147483643
   ScrollBars      =   2
   TextRTF         =   $"Simple Sockets.frx":0000
End
Begin VB.Label Label1
   Alignment       =   1  'Right Justify
   Caption         =   "Port:"
   Height          =   255
   Left            =   120
   TabIndex        =   5
   Top             =   480
   Width           =   615
End
Begin VB.Label lblSite
   Alignment       =   1  'Right Justify
   Caption         =   "Server:"
   Height          =   255
   Left            =   120
   TabIndex        =   3
   Top             =   120
   Width           =   615
End
Begin MSJSOCKLib.MSJSock MSJSockCtrl
   Height          =   375
```

```
      Left            =   120
      TabIndex        =   2
      Top             =   120
      Width           =   375
      _Version        =   65536
      _ExtentX        =   661
      _ExtentY        =   661
      _StockProps     =   0
   End
End
Attribute VB_Name = "frmSimple"
Attribute VB_Creatable = False
Attribute VB_Exposed = False


' SIMPLE SOCKETS.FRM -
'   This is the main (and only) form for the MSJ Simple Sockets program.
'   It connects to a user-supplied server and port, and allows the user to
'   interactively type commands to the server. Results are shown in a jaunty,
'   multicolored rich text edit box.


'---------------------------------------------------------------------

Private Sub btnConnect_Click()
    ' Using the MSJSock control, create a socket
    sock = MSJSockCtrl.CreateSocket

    ' Get the port the user typed in, as an integer
    port = CInt(edtPort.Text)

    ' Look up the user-supplied host name. This will come back as a
    ' GotHost event later on, at which point the connection will continue.
    host = MSJSockCtrl.gethostbyname(edtSite.Text)

    ' Add informational text to the output RTE (rich text edit), colored blue
    Call DebugToList("Connecting to host " & edtSite.Text & " port " _
                     & port & Chr$(13) & Chr$(10), RGB(0, 0, 255))
End Sub

'---------------------------------------------------------------------

Private Sub btnSend_Click()
    ' When the user clicks the send button, send their current typed-in
    ' command, with a CR-LF appended
    rc = WSSend(sock, edtOutput.Text & Chr$(13) & Chr$(10))
End Sub

'---------------------------------------------------------------------

Private Sub edtOutput_KeyPress(KeyAscii As Integer)
    ' If the user hits <enter>, treat it just as if they'd clicked the
    ' Send button.
    If KeyAscii = 13 Or KeyAscii = 10 Then
        btnSend_Click
    End If
End Sub
```

```
'---------------------------------------------------------------------

Private Sub MSJSockCtrl_connect(ByVal hTask As Long)
    Dim rcl As Long

    ' When the host connects, print an informational message to the RTE
    Call DebugToList("Connected to host." & Chr$(13) & Chr$(10), RGB(0, 0, 255))

    ' Now that you're connected, have the MSJSock control capture read and
    ' write messages
    rcl = MSJSockCtrl.AsyncSelect(sock, FD_READ + FD_WRITE + FD_CLOSE)
End Sub

'---------------------------------------------------------------------

Private Sub MSJSockCtrl_GotHost(ByVal hostent As Long)
    Dim rcl As Long

    ' Once you've found the host, tell the MSJSock control to form a
    ' connection to it using the current socket and user-indicated port.
    rcl = MSJSockCtrl.Connect(sock, port, hostent)
End Sub

'---------------------------------------------------------------------

Private Sub MSJSockCtrl_RecvData(ByVal hTask As Long, ByVal Data As String)
    Dim msg As String

    ' When the MSJSock control receives incoming data, call the WinSock
    ' recv API function to retrieve it 10k at a time.
    msg = Space(10240)
    rc = recv(sock, msg, 10240, 0)

    msg = Trim(msg)

    ' Put the received information into the RTE box using red text, so you
    ' can tell it from other messages.
    DebugToList msg, RGB(127, 0, 0)
End Sub

'---------------------------------------------------------------------

Private Sub MSJSockCtrl_SockError(ByVal hTask As Long, ByVal ENum As Integer)
    ' Print WinSock errors to the RTE using black.
    DebugToList "WinSock error " & ENum & ": " & WSErrStr(ENum) & Chr$(13) &
Chr$(10), RGB(0, 0, 0)
End Sub

'---------------------------------------------------------------------

Sub DebugToList(dbgtxt As String, clr As Long)
    Dim prvlen As Long

    ' DebugToList appends colored text to a rich text edit (RTE) box. Since
    ' you can't just do Ctrl.Text = Ctrl.Text + newtext (you'd overwrite all
    ' the old formatting, you have to move the insertion point to the end
    ' and insert new text.
```

```
    ' Find the location the new text will start (at the end of the old text)
    prvlen = Len(redInOut.Text)

    ' Move the insertion point to the end of the RTE box
    redInOut.SelStart = Len(redInOut.Text) + 1

    ' Replace the selection (which is nothing right now) with the new
    ' text, appending it without changing the previous formatting.
    redInOut.SelText = dbgtxt

    ' Now select the entire inserted text
    redInOut.SelStart = prvlen
    redInOut.SelLength = Len(dbgtxt)

    ' Change the selected (newly inserted) text to the requested color
    redInOut.SelColor = clr


    ' Move the insertion point back to the end of the RTE
    redInOut.SelStart = Len(redInOut.Text) + 1
End Sub

'--------------------------------------------------------------------

Sub HandleWSError(rc As Integer)
    ' Check for an unhandled WinSock error. If there is one, handle it
    ' just like an error coming in from the MSJSock control.
    If (rc = SOCKET_ERROR) Then
        WSErrorN = WSAGetLastError()
        Call MSJSockCtrl_SockError(-1, WSErrorN)
    End If
End Sub

'--------------------------------------------------------------------

Function WSSend(wsock As Long, strg As String) As Integer
    Dim rc As Integer

    ' As the user sends data, send it to the RTE in green.
    DebugToList strg, RGB(0, 127, 0)

    ' Call the WinSock API send function to send the data.
    rc = send(wsock, strg, Len(strg), 0)

    DoEvents

    ' Check to see if send returned an error. If so, it will be handled
    ' by HandleWSError
    HandleWSError rc
    WSSend = rc
End Function
```

Naturally, there's a lot of tracking work to be done for a full-blown news reader. However, with MSJ Simple Sockets, you can see what's actually going on behind the scenes with NNTP, or any other protocol you'd like to test out.

The Internet provides an effective solution for broadcasting information across different platforms. For example, many organizations use the Internet to distribute product information, directories, or company policy manuals to people both within and outside of the organization. By applying Internet technologies to their internal network, organizations can help their employees share, analyze, and find information more easily.

Microsoft Office 97 is a flexible and robust tool for creating Internet content. By using the Microsoft Office Internet features, you can create applications to publish and distribute information to peers, management, and other functional groups in a timely manner, regardless of where they are located. For example, you can enter data into a Microsoft Access database, and then publish that database on your company's Web server so that users on a variety of platforms can access that data with a Web browser.

In Microsoft Word, Microsoft Excel, Microsoft PowerPoint, and Microsoft Access, you can use Visual Basic for Applications to automate and extend Internet features in your custom applications.

In all likelihood, you are well aware of what the Internet is, and you've had a chance to take advantage of its many resources. Even if you have used the Internet, the following overview will help to make sure you understand the terms used to describe it in this chapter.

The *Internet* is a collection of computer networks that connects millions of computers around the world. The *World Wide Web* is a client/server technology used to access a vast variety of digital information from the Internet. By using a software client called a *Web browser*, such as Microsoft Internet Explorer, and a modem or other connection to an Internet Service Provider (ISP), you can easily access text, graphics, sound, and other digital information from practically any computer in the world that is running the appropriate server software on the Internet.

This section includes the following topics:

A Web browser uses a variety of standardized methods for addressing and communicating with Internet servers. These methods are called protocols. The most common protocol is *Hypertext Transfer Protocol* (HTTP), which was originally created to publish and view linked text documents, but has been extended to display and run a growing variety of graphics, sound, video, and other multimedia content. Other common protocols include File Transfer Protocol (FTP), Gopher, telnet, RealAudio, as well as protocols used to start other applications such as e-mail and Usenet newsreaders.

The following table describes many of the protocols commonly in use today.

Protocol	Protocol name	Description
http	Hypertext Transfer Protocol	Goes to Web pages that contain text, graphics, sound, and other digital information from a Web server on the World Wide Web.
ftp	File Transfer Protocol	Transfers files between computers on the Internet.
gopher	Gopher protocol	Displays information on a Gopher server.
wais	WAIS protocol	Accesses a Wide Area Information Servers database.
file	File protocol	Opens a file on a local hard disk or LAN.
https	Hypertext Transfer Protocol with privacy	Establishes an HTTP connection that uses Secure Sockets Layer (SSL) encryption.
mailto	MailTo protocol	Opens your electronic mail program to send a message to the specified Internet e-mail address.
msn	Microsoft Network protocol	Goes to a location on MSN, The Microsoft Network.
news	News protocol	Starts a newsreader and opens the specified Usenet newsgroup.
nntp	Network News Transfer Protocol	Performs the same function as News protocol.
mid	Musical Instrument Digital Interface (MIDI) protocol	Plays MIDI sequencer files if the user's computer has a sound card.
cid	CompuServe Dialer (CID) protocol	Establishes a point-to-point protocol (PPP) connection with the Internet through CompuServe's network.
prospero	Prospero protocol	Opens files on the Prospero distributed file system.
telnet	Telnet protocol	Starts a telnet *terminal emulation program*. A terminal emulation program is a command-line interface that you can use to issue commands on a remote computer. For example, by using telnet to connect to a UNIX server, you can issue UNIX commands to perform operations on that server.
rlogin	Rlogin protocol	Starts an Rlogin terminal emulation program.
tn3270	TN3270 protocol	Starts a TN3270 terminal emulation program.
pnm	RealAudio protocol	Plays RealAudio streaming audio from a RealAudio server. Streaming

		audio and other streaming media formats establish a connection to the server and start playing immediately without downloading an entire file.
mms	Microsoft Media Server (MMS) protocol	Plays media such as ActiveMovie streaming format files (.asf) from an MMS server.

To run or display Internet content with a Web browser, you type an address called a *Uniform Resource Locator* (URL) into its address box. You can enter a URL that points to any Internet file type or resource supported by the browser that will be used to display or run it. You enter most URLs in the following format:

protocol://serveraddress/path

Protocol specifies the Internet protocol used to establish the connection to the server, and is generally followed by a colon and two slash marks. *Serveraddress* specifies what is usually called the *domain name* of the Internet server. *Path* specifies the location and name of the page or file on the Internet server. For example, this is the URL for the What's New page on the Microsoft Access Developer Forum Web site:

http:.//www.microsoft.com/addessdev/whatsnew.htm

Note For some protocols, URLs have a different format. For example, the format for a URL that uses the MailTo protocol is **mailto:***username*@*domain*; the format for a URL that uses the News protocol is **news:***newsgroupname*; the format for a URL that uses the Network News Transfer Protocol is **nntp://***newsgroupname*.

Most files you download and open with a Web browser are pages formatted with *Hypertext Markup Language (HTML) tags*. HTML tags are codes enclosed in angle brackets that are used by a Web browser to determine the structure and appearance of an HTML document, such as graphic elements and text formatting. For example, the two HTML tags in the following sentence:

Make <B>this text</B> look bold.

Cause the text to display like this when viewed with a Web browser:

Make **this text** look bold.

To navigate to other pages or multimedia content, a user clicks a *hyperlink* on a Web page. A hyperlink is colored and underlined text, or a graphic, that uses the path specified by a URL to download and open another file, such as another Web page or some form of multimedia content, such as a picture or sound file.

You can use HTML tags called *anchors* to create hyperlinks. An anchor with an HREF attribute goes to a file outside of the current document. For example, the following anchor creates a hyperlink that goes to the Microsoft home page:

<A HREF="http://www.microsoft.com/">Microsoft Home Page</A>

An anchor with a NAME attribute creates a bookmark at a location within the same document. Other hyperlinks can go to the bookmark created with this type of anchor.

HTML was originally a simple system for publishing documents on the Web, but it's rapidly evolving to include features that you can use to create sophisticated, interactive applications.

{ewc mvimg, mvimage,!tip.bmp}

Standard Web browser functionality is evolving through the addition of a variety of new technologies such as *helper applications*, *plug-ins*, *ActiveX controls*, *Java applets*, and *scripting languages*. If your Web browser doesn't support these technologies, you may need to install additional components to be able to use them.

Helper Applications

Helper applications are typically used to play audio or video files, or to display certain graphic formats. You may need to install helper applications before you can play or display certain content in your Web browser. In more recent browsers, such as Microsoft Internet Explorer version 3.0 and Netscape Navigator version 3.0, many of these functions are built into the browser itself, or are being replaced by one of the other technologies described in this section.

Plug-Ins

By using plug-ins, Web page authors can embed content that uses additional player or reader modules directly within Web pages. For example, there are plug-ins used to display Macromedia Director and Apple QuickTime movies in a Web page. In order to use Web pages containing content that requires a plug-in, the plug-in must be installed beforehand. Microsoft Internet Explorer version 3.0 and Netscape Navigator versions 2.0 and 3.0 can run plug-ins.

ActiveX Controls

By using ActiveX controls, Web page authors can extend the kinds of content that can be displayed on a Web page. They can also enhance their Web pages with sophisticated formatting features, animation, and embedded programs that perform operations such as background downloading. ActiveX controls don't need to be installed beforehand — they can be downloaded when a user first opens a Web page. Microsoft Internet Explorer version 3.0 has built-in support for Web pages that contain ActiveX controls. To use a Web page that contains ActiveX controls in Netscape Navigator version 3.0, you must use the NCompass ScriptActive plug-in.

Java Applets

By using the Java programming language, Web page authors can produce applications called applets, which can perform functions similar to plug-ins and ActiveX controls. To display or run a Java applet from a Web page, a Web browser must be able to compile and run Java code. Microsoft Internet Explorer version 3.0 and Netscape Navigator versions 2.0 and 3.0 can run Java applets.

Scripting Languages

Scripting languages are interpreted programming languages that Web page authors can use to perform a variety of operations. They are often used in conjunction with ActiveX controls or Java applets. Three common examples of scripting languages are VBScript, JScript, and JavaScript. To use a page that contains scripting language code, a Web browser must be able to interpret the code. Microsoft Internet Explorer version 3.0 can run both VBScript and JScript code, as well as most JavaScript code. Netscape Navigator versions 2.0 and 3.0 can run JavaScript code. Netscape Navigator version 3.0 can run VBScript code if you have the NCompass ScriptActive plug-in installed.

In addition to scripting languages, there are a variety of *server-side scripting languages*, such as CGI, PERL, and ActiveX Scripting that extend the functionality of servers. In Microsoft Access, you can create Active Server Pages (ASP) that use ActiveX Scripting to bind data to Web page controls. By using Active Server Pages, your Web pages can perform many of the same functions as Microsoft Access forms.

If you install Internet server software, such as Microsoft Internet Information Server, on servers connected by a local area network (LAN), you can use these same Internet technologies to share data within an organization. Such a system is called an intranet or internal Web. For example, your organization could post human resources information for all employees on a Web page, or a project team could post information about its members and provide hyperlinks to important documentation about the project. All the features in Microsoft Office that are designed for the Internet can also be used on an intranet. For more information about Microsoft Internet Information Server, click here to connect to the IIS Web page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

You can also set up a personal web server to test your Web application or to publish small-scale intranet applications. If you are using Windows 95, you can install Microsoft Personal Web Server. If you are using Windows NT Workstation version 4.0, you can install Microsoft Peer Web Services. For more information about Personal Web Server or Peer Web Services, see Setting Up a Personal Web Server later in this chapter.

In all Microsoft Office 97 applications, you can create hyperlinks to display and run standard Internet content. Additionally, in all Office applications except Outlook, you can create hyperlinks to move between Microsoft Word documents, Microsoft Excel workbooks, Microsoft PowerPoint presentations, and Microsoft Access databases that are stored on a local hard disk or on a LAN. You don't need Internet connections or Web servers to use hyperlinks to move between Office documents or files. You can use both kinds of hyperlinks in the same application.

A hyperlink to a Microsoft Office document can also go to a specific location or object within another document or the current document. The following table lists the objects that a hyperlink can go to within each Microsoft Office application.

Application	**Object**
Microsoft Access	A table in Datasheet view. A query in Datasheet view. A form in Form view or Datasheet view, depending on the form's **DefaultView** property setting. A report in Print Preview. A macro. Using a hyperlink to go to a macro runs the macro. A module.
Microsoft Excel	A worksheet. A specified range of cells in a worksheet. A named range of cells in a worksheet.
Microsoft PowerPoint	A slide.
Microsoft Word	A bookmark.

When you follow a hyperlink, either by clicking it or by using the **Follow** or **FollowHyperlink** method, the Office application may open a cached copy of the document, depending on the Internet settings in Control Panel. To view or change these settings, double-click the **Internet** icon in Control Panel, then click **Settings** on the **Advanced** tab. For more information about the **Follow** and **FollowHyperlink** methods, see The Follow Method and The FollowHyperlink Method later in this chapter.

This section includes the following topics:

When specifying a hyperlink address, you can use either of two forms:

® A valid URL that points to a resource on the Internet or an intranet.

® A path on a local hard disk, or a path on a LAN.

This section includes the following topics:

® Specifying a URL as a Hyperlink Address

® Specifying a UNC or Standard Path as a Hyperlink Address

® Absolute vs. Relative Links

To create a hyperlink that goes to a Web page or other Internet content, you must enter a valid URL as the hyperlink address. You can enter a URL that points to any Internet file type or resource supported by the browser or to an ActiveX control, such as the WebBrowser control, that will be used to display or run it. For example, the URL to the home page of the Microsoft Office Developer Forum is:

http://www.microsoft.com/officedev/

When you enter a URL like the previous example that doesn't specify a particular file name, be sure to include a slash mark (/) at the end of the address. Although URLs that do not end in a slash mark generally work, they require the server to perform additional operations that add to the overall network load and slow down the opening of the hyperlink. When you specify a file name at the end of a URL, you do not end the URL with a slash mark. For example:

http://www.microsoft.com/default.asp

If your users have Microsoft Internet Explorer version 3.0 or if your application uses the WebBrowser control, your application can open a Microsoft Excel workbook, Word document, or PowerPoint presentation within the browser or control. To do this, the corresponding Office application or viewer (Microsoft Excel Viewer, Word Viewer, or PowerPoint Viewer) must also be installed on the user's computer. In this case, a URL can point directly to an Office document on a Web or intranet server. For example:

http://*YourIntranetServer/YourWordDoc*.doc

You can also open an Office document in Microsoft Internet Explorer version 3.0 or in the WebBrowser control directly from the standard file system, without using a Web server. To do so, use the File protocol, as follows:

file://c:\my documents\sales.doc

To create a hyperlink that starts a Microsoft Office 97 application and opens one of its documents from a LAN, enter a universal naming convention (UNC) path as the hyperlink address. This ensures that the hyperlink continues to work if the document or the application that contains the hyperlink is moved to another computer. A UNC path starts with two backslashes (\\) and supplies the server name, share name, and full path to the file. For example, a UNC path to a Microsoft Excel workbook would be in the following format:

*server**share**path**workbook*.**xls**

You can also specify a network path that uses a mapped drive letter, such as E:*path**workbook*.xls. However, because the path is specific to that drive letter, the hyperlink only works if the user's computer has the drive letter mapped to the appropriate server and share. If you want to create a hyperlink that goes to a file on a local drive, you can use a standard file path starting with a drive letter, such as C:*path**workbook*.xls. In this case, if the application is moved to another computer, the hyperlink only works if the file specified in the address is stored on the same drive and in the same folder on the new computer.

You can also enter a UNC or standard file path as a hyperlink address to open any file type that is registered on the computer running your document or application. For example, if Notepad is installed and registered to open text (.txt) files, you could enter a UNC path to open a text file in the following format:

*server**share**path**filename*.**txt**

When you create a hyperlink, you can use a path based on either an *absolute link* or a *relative link*. A path based on an absolute link points to a fixed file location. Absolute links identify the destination of a hyperlink by its full address such as C:\My Documents\Sales.doc or http:\\www.microsoft.com\default.htm. Use an absolute link for hyperlinks to destinations that won't be moved or that require a full path. For example, use absolute links in hyperlinks to other Web sites, such as a list of your favorite Web sites.

A path based on a relative link points to a destination relative to the file the hyperlink is located in. When the first part of the path is shared by both the file that contains the hyperlink and the destination file, that part is called a hyperlink base. For example, if the path to the file that contains the hyperlink is C:\My Documents\Databases and the path to the destination file is C:\My Documents\Workbooks, then C:\My Documents is the *hyperlink base*. The hyperlink base address is automatically added to the beginning of the path for all relative links. You can specify the hyperlink base on the **Summary** tab of the document's property sheet. To open a document's property sheet in Microsoft Excel, Word, or PowerPoint, click **Properties** on the File menu. To open the property sheet for a Microsoft Access database, click **Database Properties** on the **File** menu.

When a hyperlink uses a path based on a relative link, you can move the file that contains the hyperlink and the destination file without breaking the hyperlink if you move the destination file to an identically named location. For example, if you set the hyperlink base to C:\My Documents and then create a relative link to a document in C:\My Documents\Workbooks, and you move the document that contains the relative link to a new computer, you must copy the destination file into an identically named folder in C:\My Documents on the new computer. Alternatively, you can move the destination file to a folder named Workbooks within another folder (for example, D:\Applications), and then open the document that contains the relative link and update the hyperlink base to the new folder's name.

If you save a document or database object that contains relative links as an HTML document, the hyperlink base is omitted from the anchor tags created for those relative links. For example, suppose you create a relative link to a database called Names.mdb in C:\My Documents\Databases, and set the hyperlink base to C:\My Documents. When you save the document as an HTML document, the anchor tag created is `<A HREF="Databases\Names.mdb">`. To keep the hyperlink from breaking in the HTML document, you must create a Databases folder in the folder that contains the HTML document on the HTTP server, and then copy the Names.mdb database into that folder.

All Office 97 applications except Outlook provide objects and collections that you can use to work with hyperlinks in Visual Basic code. Although there are a great number of similarities across each application, in some cases the objects and collections that hyperlinks are associated with differ slightly in each Office application.

To work with hyperlinks in Visual Basic code, you use the **Hyperlink** object. In all Office 97 applications except Microsoft Access, the **Hyperlink** object is a member of the **Hyperlinks** collection. In Microsoft Access, the **Hyperlink** object is a member of the **Controls** collection. The objects that can contain a **Hyperlinks** collection differ for each application. They are listed in the following table.

Application	**Objects that can contain a Hyperlinks collection**
Microsoft Word	**Document**, **Range**, or **Selection** objects
Microsoft Excel	**Worksheet** or **Range** objects
Microsoft PowerPoint	**Slide** or **Master** objects
Microsoft Access	None. Microsoft Access doesn't have a **Hyperlinks** collection. All **Hyperlink** objects are members of the **Controls** collection. In addition, you can have a set of records that contains fields with the Hyperlink data type and use Visual Basic to work with the records as if they were a collection.

The objects that can have a **Hyperlink** object associated with them differ for each application. The following table summarizes which objects can have an associated **Hyperlink** object.

Application	**Objects that can have a Hyperlink object associated with them**
Microsoft Word	**Shape**, **InlineShape**, **Selection**, or **Range** objects
Microsoft Excel	**Shape**, **Selection**, or **Range** objects
Microsoft PowerPoint	**Shape.ActionSettings** or **TextRange.ActionSettings** objects
Microsoft Access	**CommandButton**, **ComboBox**, **Image**, **Label**, **ListBox**, or **TextBox** objects

This section includes the following topics:

In Microsoft Excel and Word, use the **Add** method to create a **Hyperlink** object and add it to the **Hyperlinks** collection. To create a hyperlink with the **Add** method, use the following syntax:

object.**Add**(anchor, address, subaddress)

The following table describes the arguments of the **Add** method.

Argument	**Description**
object	Required. An expression that returns a **Hyperlink** object.
anchor	Required. The anchor for the hyperlink. Can be either a **Range** or a **Shape** object.
address	Required. The address of the hyperlink.
subaddress	Optional. The subaddress of the hyperlink.

This section includes the following topics:

® Microsoft Word Examples

® Microsoft Excel Examples

® Microsoft PowerPoint Examples

® Microsoft Access Example

Use the **Hyperlink** property to return a reference to a **Hyperlink** object. The objects that can have a **Hyperlink** object associated with them differ somewhat for each application.

Microsoft Word Example

Use the **Hyperlink** property to return the hyperlink for a shape. Note that a shape can have only one hyperlink associated with it. The following example follows the hyperlink associated with the first shape in the active document.

```
ActiveDocument.Shapes(1).Hyperlink.Follow
```

Selection and **Range** objects can have multiple **Hyperlink** objects associated with them. For these objects, you must either loop through the object's **Hyperlinks** collection or specify a member of the object's **Hyperlinks** collection by using the **Item** method. The following example loops through the hyperlinks in the current selection.

```
Dim H As Hyperlink, hLinks As Hyperlinks
Set hLinks = Selection.Hyperlinks

For Each H In hLinks
    MsgBox H.Address
Next H
```

The following example displays the address of the first hyperlink in the first 20 characters of the current document in the Immediate window.

```
Debug.Print ActiveDocument.Range(0,20).Hyperlinks(1).Address
```

Microsoft Excel Example

Use the **Hyperlink** property to return the hyperlink for a shape. Note that a shape can have only one hyperlink associated with it. The following example follows the hyperlink associated with the first shape on the first worksheet.

```
Worksheets(1).Shapes(1).Hyperlink.Follow NewWindow:=True
```

Microsoft PowerPoint Example

As mentioned earlier in this chapter, a shape in PowerPoint can have up to two different hyperlinks assigned to it: one that's followed when the user clicks the shape during a slide show, and another that's followed when the user passes the mouse pointer over the shape during a slide show. To return a hyperlink for a shape, you must first reference the appropriate member of the **ActionSettings** collection (**ppMouseOver** or **ppMouseClick**), and then use the **Hyperlink** property.

The following example displays the address for the mouse-click hyperlink of the third shape on the first slide of the active presentation in the Immediate window.

```
Debug.Print ActivePresentation.Slides(1).Shapes(3). _
    ActionSettings(ppMouseClick).Hyperlink.Address
```

Microsoft Access Example

In Microsoft Access, you can use the **Hyperlink** property to return a reference to the **Hyperlink** object associated with a **CommandButton**, **ComboBox**, **Image**, **Label**, **ListBox**, or **TextBox** control.

The CreateHyperlink procedure in the following example sets the **Address** and **SubAddress** properties for a label, image control, or command button to the values passed to the procedure. The **Address** property setting is optional, because a hyperlink to a database object in the current database uses only the **SubAddress** property.

To try this example, create a form with two text box controls named txtAddress and txtSubAddress, and a command button named cmdFollowLink. Then paste the sample code into the Declarations section of the form's module. Display the form in Form view, enter appropriate values in the txtAddress and txtSubAddress text

boxes, and click the cmdFollowLink button.

```
Private Sub cmdFollowLink_Click()
    CreateHyperlink Me!cmdFollowLink, Me!txtSubAddress, Me!txtAddress
End Sub

Sub CreateHyperlink(ctlSelected As Control, txtSubAddress As TextBox, _
    Optional txtAddress As TextBox)
    Dim hlk As Hyperlink

    Select Case ctlSelected.ControlType
        Case acLabel, acImage, acCommandButton
            Set hlk = ctlSelected.Hyperlink
            With hlk
                If Not IsMissing(txtAddress) Then
                    .Address = txtAddress
                Else
                    .Address = ""
                End If
                .SubAddress = txtSubAddress
                .Follow
                .Address = ""
                .SubAddress = ""
            End With
        Case Else
            MsgBox "The control '" & ctlSelected.Name & "' does not support
hyperlinks."
    End Select
End Sub
```

Use the **Item** method (or the **Item** property in Microsoft Excel) of the **Hyperlinks** collection to return a single **Hyperlink** object based on its position in the collection. The first object in the collection has an **Item** value of 1. The **Item** method is the default member of the **Hyperlinks** collection, so you can refer to the **Item** method in either of the following ways:

```
Hyperlinks.Item(1)
Hyperlinks(1)
```

Microsoft Word Example

The following example follows the first hyperlink in the selection.

```
If Selection.Hyperlinks.Count >= 1 Then
    Selection.Hyperlinks(1).Follow
End If
```

Note The Count property for the **Hyperlinks** collection of a **Selection** object returns the number of items in the main story only. To count items in other stories, specify the story in the **StoryRanges** collection. For example, to count all of the hyperlinks in the primary footer story you can use the following code.

```
ActiveDocument.StoryRanges(wdPrimaryFooterStory).Hyperlinks.Count.
```

Microsoft Excel Example

The following example uses the Follow method to activate the second hyperlink in the range of cells from E5 to E8.

```
Worksheets(1).Range("E5:E8").Hyperlinks(2).Follow
```

Microsoft PowerPoint Example

The following example sets the Address property of the second hyperlink on the first slide in the current PowerPoint presentation.

```
ActivePresentation.Slides(1).Hyperlinks(2).Address = "C:\New\Newsales.ppt"
```

You can use the **Hyperlinks** collection in Microsoft Excel, Word, and PowerPoint to loop through the set of **Hyperlink** objects associated with an object. In Microsoft Access, you can loop through the **Controls** collection or a set of records to work with the hyperlinks in your application.

The following examples perform operations on a **Hyperlinks** collection that contains existing **Hyperlink** objects. In the Microsoft Excel, Word, and PowerPoint examples that follow, the object that contains the **Hyperlinks** collection is specific to the application. However, you can modify each example to run in another application by referring to the appropriate object. Because the Microsoft Access examples use the **Controls** collection or a set of records instead of the **Hyperlinks** collection, you can only use them in Microsoft Access.

Microsoft Word Example

If the active document includes hyperlinks, this example inserts a list of the hyperlink destinations at the end of the document.

```
Dim hLink As Hyperlink

Set myRange = ActiveDocument.Range(Start:=ActiveDocument.Content.End - 1)
Count = 0
For Each hLink In ActiveDocument.Hyperlinks
    Count = Count + 1
    With myRange
        .InsertAfter "Hyperlink #" & Count & vbTab
        .InsertAfter hLink.Address
        .InsertParagraphAfter
    End With
Next hLink
```

Microsoft Excel Example

The following example updates all hyperlinks on the first worksheet in the active workbook that have the specified address.

```
Dim hLink As Hyperlink

For Each hLink in ActiveWorkbook.Sheets(1).Hyperlinks
    If LCase(hLink.Address) = "C:\Current Work\Sales.ppt" Then
        hLink.Address = "C:\New\Newsales.ppt"
    End If
Next hLink
```

Note In Word, you can use the **Hyperlinks** collection to access hyperlinks created by inserting a HYPERLINK field. In Microsoft Excel, however, you cannot use the **Hyperlinks** collection to access hyperlinks created by entering a formula using the HYPERLINK function.

Microsoft PowerPoint Example

The following example updates an outdated Internet address for all hyperlinks in the active presentation.

```
Dim hLink As Hyperlink
Dim S As Slide

oldAddr = InputBox("Old internet address")
newAddr = InputBox("New internet address")
For Each S In ActivePresentation.Slides
    For Each hLink In s.Hyperlinks
        If LCase(hLink.Address) = Lcase(oldAddr) Then hLink.Address = newAddr
    Next hLink
```

```
Next S
```

Microsoft Access Examples

Microsoft Access doesn't support the **Hyperlinks** collection, but you can loop through the **Controls** collection on a form or report to work with the hyperlinks associated with any control on the form or report. The following procedure displays the name and hyperlink address values for controls that contain hyperlinks in the Debug window.

```
Sub ListHyperlinks(strForm As String)
    Dim Frm As Form
    Dim Ctl As Control

    DoCmd.OpenForm strForm, acDesign, , , , acHidden
    Set Frm = Forms(strForm)

    ' Ignore controls without hyperlinks.
    On Error Resume Next

    For Each Ctl In Frm.Controls
        If Not (Ctl.ControlType = acTextBox) Then
        Debug.Print "Control:" & Ctl.Name & vbCrLf & _
            "Address:" & Ctl.Hyperlink.Address & vbCrLf & _
            "Subaddress:" & Ctl.Hyperlink.SubAddress & vbCrLf
        Else
        Debug.Print "Control:" & Ctl.Name & vbCrLf & _
            "Text box control bound to Hyperlink field " & _
            Ctl.ControlSource & vbCrLf
        End If
    Next Ctl
    Frm.Close
End Sub
```

In addition to creating **Hyperlink** objects that belong to the **Controls** collection of forms and reports, you can have a set of records that contains fields with the Hyperlink data type and use Visual Basic to work with the records as if they were a collection. For example, you can loop through the records in a table to work with the properties of a field. The following procedure works with the Suppliers table in the Northwind sample database. If a field is a Hyperlink field, the procedure loops through all the records in the table. If a field is not null (empty), it displays the record number, field name, and displayed value in the Debug window.

```
Sub HyperlinkRecordset()
    Dim dbs As Database
    Dim rstSuppliers As Recordset
    Dim fldField As Field

    ' Return reference to current database.
    Set dbs = CurrentDb
    ' Create dynaset-type Recordset object.
    Set rstSuppliers = dbs.OpenRecordset("Suppliers", dbOpenDynaset)

    ' Print displayed value for fields containing hyperlinks.
    For Each fldField In rstSuppliers.Fields
        If (fldField.Attributes And dbHyperlinkField) Then
            With rstSuppliers
                Do While Not .EOF
                    If Not IsNull(fldField.Value) Then
                        Debug.Print rstSuppliers.AbsolutePosition + 1 & " " & _
                        fldField.Name & " " & _
                        HyperlinkPart(fldField.Value, acDisplayedValue)
```

```
                    End If
                .MoveNext
                Loop
            .MoveFirst
            End With
        End If
    Next fldField
    ' Free all object variables.
    rstSuppliers.Close
    Set dbs = Nothing
End Sub
```

For more information about Hyperlink fields, see Storing Hyperlinks in Microsoft Access Tables later in this chapter.

The following table summarizes the methods and properties you can use to work with hyperlinks in Visual Basic.

Method or property name	Description
Hyperlink property	Returns a reference to a hyperlink object in code.
Follow method	Follows a hyperlink defined by an existing **Hyperlink** object. The Follow method has the same effect as clicking the hyperlink.
FollowHyperlink method	Follows a hyperlink address specified in code or passed to the method from a text box. For example, you can prompt a user to type a hyperlink address in a dialog box or form, and then use the **FollowHyperlink** method to go to that address.
ExtraInfoRequired property (Word only)	A read-only property that returns **True** if extra information is required to resolve the specified hyperlink. You can specify extra information, such as a file name or a query string, by using the extrainfo argument with the **Follow** or **FollowHyperlink** methods.
AddToFavorites method	Adds a shortcut to the Favorites folder. The **AddToFavorites** method can reference a **Hyperlink** object or the current document (Microsoft Access database, Microsoft Excel workbook, Microsoft PowerPoint presentation, or Microsoft Word document).
Address property	Returns the address of the specified hyperlink. This property is read/write, except in Word, where it is read-only.
Subaddress property	Returns a named location in the destination of the specified hyperlink. The named location can be a bookmark (Microsoft Word), a named cell or cell reference (Microsoft Excel), a database object (Microsoft Access), or a slide number (Microsoft PowerPoint). This property is read/write, except in Word, where it is read-only.
Type property (Microsoft Excel, Word, and PowerPoint only)	Returns the type of object the hyperlink is associated with. Can be one of the following constants: **msoHyperlinkInlineShape** (Word only) **msoHyperlinkRange** **msoHyperlinkShape**
HyperlinkAddress property (Microsoft Access only)	Sets or returns the address of a hyperlink for a label, image control, or command button. The **HyperlinkAddress** property is equivalent to setting or returning the Address property for the control in Visual Basic; for example, *object*.**HyperlinkAddress** is equivalent to *object*.**Hyperlink.Address**. You can also set the **HyperlinkAddress** property in the control's property sheet.
HyperlinkSubAddress property (Microsoft Access only)	Sets or returns the location within the Office document or object specified by the **HyperlinkAddress** property. When no **HyperlinkAddress** property is specified, **HyperlinkSubAddress** specifies a database object in the current database. The **HyperlinkSubAddress** property is equivalent to setting or returning the **SubAddress** property for

	the control in Visual Basic; for example, object.**HyperlinkSubAddress** is equivalent to object.**Hyperlink.SubAddress**. You can also set the **HyperlinkSubAddress** property in the control's property sheet.

For more information about these methods and properties, search Help in the appropriate application for the name of the method or property.

This section includes the following topics:

® The Follow Method

® The FollowHyperlink Method

® Handling Hyperlink Errors

® The AddToFavorites Method

The **Follow** method follows a hyperlink defined by an existing **Hyperlink** object, and has the same effect as clicking the hyperlink. The **Follow** method downloads the document or Web page specified by the hyperlink address associated with a **Hyperlink** object and opens it in the appropriate application. If the hyperlink refers to a file system path or uses the File protocol, the **Follow** method opens the document instead of downloading it.

The syntax for the **Follow** method is:

expression.**Follow**(*newwindow*, *addhistory*, *extrainfo*, *method*, *headerinfo*)

The following table describes the arguments of the Follow method.

Argument	Description
expression	Required. An expression that returns a **Hyperlink** object.
newwindow	Optional. A **Boolean** value where **True** (-1) opens the document in a new window and **False** (0) opens the document in the current window. The default value is **False**.
addhistory	Optional. A **Boolean** value where **True** (-1) adds the hyperlink to the History folder and **False** (0) doesn't add the hyperlink to the History folder. The default value is **True**.
extrainfo	Optional. A string or an array of **Byte** data that specifies additional information for HTTP to use to resolve the hyperlink. For example, you can use the *extrainfo* argument to specify the coordinates of an image map or the contents of a form. The string is either appended or posted, depending on the value of the *method* argument. In Word, you can use the **ExtraInfoRequired** property to determine whether extra information is required.
method	Optional. Specifies the way the *extrainfo* argument is handled. You can set the *method* argument to **msoMethodGet** or **msoMethodPost**.
headerinfo	Optional. A string that specifies header information for the HTTP request. The default value is a zero-length string (" "). You can combine several header lines into a single string by using the following syntax: "*string1*" **& vbCr &** "*string2*" The specified string is automatically converted into ANSI characters. Note that the *headerinfo* argument may overwrite default HTTP header fields.

For the *method* argument of the **Follow** method, you can specify one of the constants described in the following table.

Constant	Description
msoMethodGet	The *extrainfo* argument is a string that's appended to the URL, separated by a question mark, when you use the HTTP GET method from an HTML form. For example, you can submit a query to an HTTP server by using an address in the following format: http://www.web.com/cgi-bin/srch?*item1+item2* *item1+item2* is the extra information that's passed to the srch program on the HTTP server.
msoMethodPost	The *extrainfo* argument is posted to the server as a string or a byte array when you use the HTTP POST method. For example, data from a form is typically submitted to an HTTP server with a series of name/value pairs in the following format: *name1=value1***&***name2=value2* This data can be submitted as either a string or byte array,

depending on what format the program on the server has been programmed to use. Use the HTTP POST method to submit extra information if the program on the HTTP server is reading the form's data from the standard input stream (STDIN).

For examples that illustrate uses of the **Follow** method, see the code samples in previous sections of this chapter.

The **FollowHyperlink** method follows a hyperlink address specified in code or passed to the method from a variable or object. For example, you can prompt a user to type a hyperlink address in a dialog box, and then use the **FollowHyperlink** method to go to that address. The **FollowHyperlink** method downloads the document or Web page specified by the hyperlink address associated with a **Hyperlink** object and opens it in the appropriate application. If the address refers to a file system path or uses the File protocol, the **FollowHyperlink** method opens the document instead of downloading it.

The syntax for the FollowHyperlink method is:

expression.**FollowHyperlink**(*address*, *subaddress*, *newwindow*, *addhistory*, *extrainfo*, *method*, *headerinfo*)

The following table describes the arguments of the FollowHyperlink method.

Argument	Description
expression	Required. An expression that returns one of the following objects: Microsoft Word **Document** object Microsoft Excel **Workbook** object Microsoft PowerPoint **Presentation** object Microsoft Access **Application** object
address	A string expression that evaluates to a valid hyperlink address.
subaddress	A string expression that evaluates to a named location in the document specified by the *address* argument. The default is a zero-length string (" "). If no address is specified, *subaddress* specifies a named location in the document or database.

For information about the *newwindow*, *addhistory*, *extrainfo*, *method*, and *headerinfo* arguments, see the preceding section, The Follow Method.

Microsoft Word Examples

This example follows the specified URL and displays the Microsoft home page in a new window.

```
ActiveDocument.FollowHyperlink Address:="http://www.microsoft.com", _
    NewWindow:=True, AddHistory:=True
```

This example opens the HTML document named Default.htm directly from the local hard disk.

```
ActiveDocument.FollowHyperlink Address:="file://C:\Pages\Default.htm"
```

Microsoft Excel Example

This example follows the specified URL address and displays the names of all the topics related to opera.

```
ActiveWorkbook.FollowHyperlink Address:="http://search.Yahoo.com/bin/search", _
    AddHistory:=False, Method:=msoMethodGet, ExtraInfo:="p=Opera"
```

Microsoft PowerPoint Example

This example loads the document at www.gohere.com in a new window and adds it to the History folder.

```
Application.ActivePresentation.FollowHyperlink _
    Address:="http://www.gohere.com", NewWindow:=True, AddToHistory:=True
```

Microsoft Access Examples

The following function prompts a user for a hyperlink address and then follows the hyperlink.

```
Function GetUserAddress() As Boolean
    Dim strInput As String
```

```
    On Error GoTo Error_GetUserAddress
    strInput = InputBox("Enter a valid address")
    Application.FollowHyperlink strInput, , True
    GetUserAddress = True

Exit_GetUserAddress:
    Exit Function

Error_GetUserAddress:
    MsgBox Err & ": " & Err.Description
    GetUserAddress = False
    Resume Exit_GetUserAddress
End Function

You can call this function with a procedure such as the following.
Sub CallGetUserAddress()
    If GetUserAddress = True Then
        MsgBox "Successfully followed hyperlink."
    Else
        MsgBox "Could not follow hyperlink."
    End If
End Sub
```

In Microsoft Access, you can also use the **FollowHyperlink** method to specify a hyperlink for controls that don't support the **HyperlinkAddress** or **HyperlinkSubAddress** properties (controls other than labels, image controls, and command buttons, or text boxes bound to Hyperlink fields).

This example uses the **FollowHyperlink** method to add hyperlink behavior to an unbound object frame control. Add the following code to the Click event of an unbound object frame named OLEUnbound1 to start a Web browser and open the specified hyperlink address when you click the image.

Note You can use similar code in Microsoft Excel, Word, or PowerPoint to create a command button that follows a hyperlink. To do so, add a command button by using the **Control Toolbox**, and then define a Click event procedure for the button. For more information, see "Creating a Hyperlink Associated with a Command Button" in Microsoft Word Examples, earlier in this chapter.

```
Private Sub OLEUnbound1_Click()
    Dim strAddress As String

    On Error GoTo Error_OLEUnbound1

    ' Set reference to hyperlink address.
    strAddress = "http://www.microsoft.com/"

    ' Follow hyperlink address.
    Application.FollowHyperlink strAddress, , True

Exit_OLEUnbound1:
    Exit Sub

Error_OLEUnbound1:
    MsgBox Err & ": " & Err.Description
    Resume Exit_OLEUnbound1
End Sub
```

{ewc mvimg, mvimage,!tip.bmp}

If an error occurs when using the **Follow** or **FollowHyperlink** methods in Visual Basic, an Automation error is displayed that contains only an error number in both decimal and hexadecimal format. For example, if xyz.htm doesn't exist, and you run the following code in Microsoft Access:

```
Application.FollowHyperlink "http://www.microsoft.com/xyz.htm"
```

the error message shown in the following illustration occurs.

{ewc mvimg, mvimage,!illust.bmp}

This error number indicates that the requested item could not be found.

You can prevent these error messages from being displayed to users of your application. To do so, check the **Number** property of the **Err** object against a decimal value in the table that follows. Then handle the error by either returning an appropriate message or performing a suitable action. If you want to be certain that all errors are handled, write an error handler that traps the entire set of error numbers.

```
Function GetUserAddress() As Boolean
    Dim strInput As String
    Dim lngErrNumber As Long

    On Error GoTo Error_GetUserAddress
    strInput = InputBox("Enter a valid address")
    Application.FollowHyperlink strInput, , True
    GetUserAddress = True

Exit_GetUserAddress:
    Exit Function

Error_GetUserAddress:
    ' Set variable equal to error number.
    lngErrNumber = Err.Number
    ' Check variable against all possible error numbers.
    Select Case lngErrNumber
        Case -2146697211
            MsgBox "Cannot locate the Internet server or proxy server."
        Case -2146697210
            MsgBox "The site reports that the item you requested " _
            & "could not be found. (HTTP/1.0 404)"
        .
        . ' Repeat for all possible error numbers.
        .
    End Select
    GetUserAddress = False
    Resume Exit_GetUserAddress
End Function
```

The following table lists the error numbers and descriptions for all errors that can occur when using the Follow and FollowHyperlink methods.

Decimal error number	Hexadecimal error number	Description
-2146697214, -2147221020 and -2147012891	0x800C0002, 0x800401E4 and 0x80072EE5	The address of this site is not valid. Check the address and try again.
-2146697213	0x800C0003	Cannot start an Internet session.
-2146697212 and -2147012867	0x800C0004 and 0x80072EFD	Cannot connect to the Internet server.
-2146697211	0x800C0005	Cannot locate the Internet server or

		proxy server.
-2146697210 and -2147012868	0x800C0006 and 0x80072EFC	The site reports that the item you requested could not be found. (HTTP/1.0 404)
-2146697209	0x800C0007	The Internet site reports that a connection was established but the data is not available.
-2146697208	0x800C0008	Cannot download the information you requested.
-2146697207	0x800C0009	The item you requested requires proper authentication. (HTTP/1.0 401)
-2146697206	0x800C000A	The Internet site cannot return the object you requested. (HTTP/1.0 403)
-2146697205 and -2147012894	0x800C000B and 0x80072EE2	The connection to this Internet site took longer than the allotted time.
-2146697204	0x800C000C	The site reports that the request is not valid.
-2146697203 and -2147012888	0x800C000D and 0x80072EE8	The required Internet protocol is not installed on your computer, or the Internet address you requested may not be valid.
-2146697202	0x800C000E	A security problem has occurred.
-2146697201 and -2147221014	0x800C000F and 0x800401EA	Cannot open the specified file.
-2146697200	0x800C0010	Cannot start the program needed to open this file.
-2147221018 and -2147221164	0x800401E6 and 0x80040154	No program is registered to open this file.
-2147467260	0x80004004	The hyperlink cannot be followed to the destination.

Note In PowerPoint, one error number is returned for all **Follow** or **FollowHyperlink** method errors: -2147467259 (0x80004005).

The **AddToFavorites** method adds a shortcut to the Favorites folder in the Windows program folder.

The syntax for the **AddToFavorites** method is:

expression.**AddToFavorites**

Expression is an expression that returns either a **Hyperlink** object or one of the objects listed in the following table.

Application	Object
Microsoft Word	**Document**
Microsoft Excel	**Workbook**
Microsoft PowerPoint	**Presentation**
Microsoft Access	**Application**

When referring to a **Hyperlink** object, the shortcut is the friendly name of the document. The friendly name is determined by the text in the <TITLE> HTML tag. If the document doesn't have a friendly name, the shortcut name is resolved by the application. If there is an existing shortcut of the same name, it is overwritten without notification.

Microsoft Word Examples

In Word, the shortcut created by the **AddToFavorites** method can refer to a **Document** object or to a **Hyperlink** object.

The following example creates a shortcut to Sales.doc and adds it to the Favorites folder. If Sales.doc isn't currently open, Word opens it from the C:\My Documents folder.

```
Sub AddDocument()
    Dim isOpen As Boolean, doc As Document

    For Each doc In Documents
        If LCase(doc.Name) = "Sales.doc" Then isOpen = True
    Next doc
    If isOpen <> True Then Documents.Open _
        FileName:="C:\My Documents\Sales.doc"
    Documents("Sales.doc").AddToFavorites
End Sub
```

To add an existing hyperlink in the document to the Favorites folder, you must refer to the document's **Hyperlinks** collection. The following example adds all of the hyperlinks in the document to the Favorites folder.

```
Sub AddHyperlinks()
    Dim H As Hyperlink
    Dim Hlinks As Hyperlinks

    Set Hlinks = ActiveDocument.Hyperlinks

    For Each H In Hlinks
        H.AddToFavorites
    Next H
End Sub
```

Microsoft Excel Examples

In Microsoft Excel, the shortcut created by the **AddToFavorites** method can refer to a **Workbook** object or to a **Hyperlink** object.

To create a shortcut to the current workbook and add it to the Favorites folder, use the following code.

```
ActiveWorkbook.AddToFavorites
```

To add an existing hyperlink in the current workbook to the Favorites folder, you must refer to the workbook's Hyperlinks collection. For example, to create a shortcut to the address in the first hyperlink in the active workbook and add it to the Favorites folder, use the following code.

```
ActiveWorkbook.Sheets(1).Hyperlinks(1).AddToFavorites
```

Microsoft PowerPoint Examples

In PowerPoint, the shortcut created by the **AddToFavorites** method can refer to a **Presentation** object or to a **Hyperlink** object.

To add a shortcut to the current presentation, use the following code.

```
Application.ActivePresentation.AddToFavorites
```

To add a hyperlink in the current slide to the Favorites folder, you must refer to the slide's **Hyperlinks** collection. The following example adds all of the hyperlinks on the first slide of the current presentation to the Favorites folder.

```
Sub AddHyperlinks()
    Dim H As Hyperlink
    Dim Hlinks As Hyperlinks

    Set Hlinks = ActivePresentation.Slides(1).Hyperlinks

    For Each H In Hlinks
        H.AddToFavorites
    Next H
End Sub
```

Microsoft Access Examples

In Microsoft Access, the shortcut created by the **AddToFavorites** method can refer to the **Application** object, which represents the current database, or to a hyperlink associated with a **Control** object.

To create a shortcut to the current database and add it to the Favorites folder, use the following code.

```
Application.AddToFavorites
```

To refer to a hyperlink associated with a **Control** object, you must use the **Hyperlink** property to access the **Hyperlink** object. For example, to create a shortcut to a hyperlink defined for a command button named Command0 on the current form and add it to the Favorites folder, use the following code.

```
Me!Command0.Hyperlink.AddToFavorites
```

In Microsoft Access 97, you can create a field with the Hyperlink data type to store hyperlink addresses in a table. You can follow a hyperlink stored in a Microsoft Access table by clicking it in the table. However, typically the field is bound to a text box, list box, or combo box control on a form. Like other bound fields, as the user moves from record to record, the value in the control changes to display the current record's hyperlink value. For example, you can use hyperlinks in this way to create an application in which users can go to Web pages, or to other content on the Internet or an intranet, from a predefined list of addresses.

In addition to storing hyperlinks to Internet addresses, you can also use hyperlinks in Microsoft Access to go to database objects and other Office documents. For example, you could create a document management application that uses a Hyperlink field to store paths to Word documents on a network. Users of such an application could add records to track new documents, or click the hyperlink in a previously added record to open the specified document.

To create a Hyperlink field, add a field in table Design view and set its **DataType** property to Hyperlink. You can also create a Hyperlink field in table Datasheet view by clicking Hyperlink **Column** on the **Insert** menu.

This section includes the following topics:

In Microsoft Access, a Hyperlink field stores up to three pieces of information: the *displaytext*, the *address*, and the *subaddress*. Each piece is separated by a pound sign (#), in the following format:

displaytext#*address*#*subaddress*

The following table describes each piece of the Hyperlink field storage format.

Piece	Description	Required?
displaytext	The text the user sees in the Hyperlink field in a table, or in a text box bound to the Hyperlink field. You can set the display text to any text string. For example, you may want the display text to be a descriptive name for the Web site or object specified by the *address* and *subaddress*. If you do not specify display text, Microsoft Access displays the value of *address*, or *subaddress* if *address* is also not specified.	No
address	A valid URL that points to a page or file on the Internet or an intranet, or the path to a file on a local hard disk or LAN. If you enter a path on a LAN, you can omit a mapped drive letter and use the universal naming convention (UNC) format: *server**share*\ *path**filename*. This prevents the path from becoming invalid if the database is later copied to another computer's hard disk or into a shared network folder.	Yes, unless *subaddress* points to an object in the current database (.mdb) file.
subaddress	The location within a file or document; for example, a database object, such as a form or report. When referring to a database object, the name of the object should be preceded by its type: Table, Query, Form, Report, Macro, or Module. Other possible values for *subaddress* include a bookmark in a Word document, a NAME anchor tag in an HTML document, a PowerPoint slide, or a cell in a Microsoft Excel worksheet.	No

Each piece of the Hyperlink field storage format can be up to 2,000 characters. The maximum length of the entire Hyperlink field value is 6,000 characters.

The following table gives examples of valid Hyperlink field values.

Hyperlink field value	Goes to
Cajun Delights#http://www.cajundelights.com/cajun.htm#	The Cajun Delights Web page. Only the words "Cajun Delights" are displayed in the field or control.
#http://www.cajundelights.com/cajun.htm#	The Cajun Delights Web page. The text "http://www.cajundelights.com" appears in the field or control because no display text is specified.
#http://www.cajundelights.com/cajun.htm#Price	The HTML anchor with the NAME attribute Price on the Cajun Delights Web page. The text "http://www.cajundelights. com/cajun.htm" is displayed.
Resume#c:\windows\personal\resume.doc#	A Microsoft Word file named Resume.doc located in the \Windows\Personal folder. Only the word "Resume" is displayed in the field or

	control.
#c:\windows\personal\resume.doc#	A Microsoft Word file named Resume.doc located in the \Windows\Personal folder. The text “c:\windows\personal\resume.doc” appears in the field or control because no display text is specified.
#c:\windows\personal\resume.doc#Qualifications	The section in the Resume.doc Word file marked with the bookmark name Qualifications. The text “c:\windows\personal\resume.doc” is displayed.
#\\databases\samples\northwind.mdb#Form Suppliers	The Suppliers form in the Northwind sample application located in the Samples share on the Databases server on a LAN (UNC format path). The text “\\databases\samples\northwind.mdb” is displayed.
Suppliers Form##Form Suppliers	The Suppliers form in the current database. The words “Suppliers Form” are displayed in the field or control.
#c:\windows\personal\1996 Sales.ppt#13	Slide 13 in the 1996 Sales PowerPoint presentation located in the \Windows\Personal folder. The text “c:\windows\personal\1996 Sales.ppt” is displayed.
#c:\windows\personal\budget.xls #Sheet1!A2	The A2 cell in Sheet1 of the Budget.xls file located in the \Windows\Personal folder. The text “c:\windows\personal\budget.xls” is displayed.

You can enter data in a Hyperlink field in three ways: by using the **Insert Hyperlink** dialog box (available through the **Hyperlink** command on the **Insert** menu), by typing an address directly into a Hyperlink field, or by using Data Access Objects (DAO) methods in Visual Basic code. When you use the **Insert Hyperlink** dialog box or type directly into a Hyperlink field, Microsoft Access adds the two pound signs (#) that delimit the parts of the hyperlink data. When you use DAO methods, your code must include the two pound signs to delimit the parts of the hyperlink data.

You can display the stored hyperlink format in a table by moving the insertion point into a Hyperlink field using the keyboard, and then pressing F2. You can edit the stored hyperlink in this form, but be careful to enter pound signs in the appropriate locations. You can add or edit the displaytext part of a Hyperlink field by right-clicking a hyperlink in a table, pointing to **Hyperlink** on the shortcut menu, and then typing the display text in the **Display Text** box. You can add or edit the address or subaddress part of a Hyperlink field by right-clicking a hyperlink in a table, pointing to **Hyperlink** on the shortcut menu, and then selecting **Edit Hyperlink**.

The **HyperlinkPart** function returns information about data stored in a **Hyperlink** field. The syntax for the **HyperlinkPart** function is:

object.**HyperlinkPart**(hyperlink **As Variant**, part **As Integer**)

The following table describes the arguments of the **HyperlinkPart** function.

Argument	Description
object	Optional. The **Application** object.
hyperlink	Required. A **Variant** that represents the data stored in a Hyperlink field.
part	Optional. The value for the part argument is an intrinsic constant that represents the information you want returned by the **HyperlinkPart** function.

You can set the part argument to the following constants.

Constant	Value	Description
acDisplayedValue	0	(Default) The underlined text displayed in a hyperlink.
acDisplayText	1	The *displaytext* part of a Hyperlink field.
acAddress	2	The *address* part of a Hyperlink field.
acSubAddress	3	The *subaddress* part of a Hyperlink field.

Note If you use the **HyperlinkPart** function in an SQL statement or a query, the part argument is required and you can't set it to the constants listed in the preceding table — you must use the value instead.

You use the **HyperlinkPart** function to return one of three values stored in a Hyperlink field (displaytext, address, or subaddress) or the displayed value. The value returned depends on the setting of the part argument. If you don't use the part argument, the **HyperlinkPart** function returns the value Microsoft Access displays for the hyperlink (which corresponds to the **acDisplayedValue** setting for the part argument).

When a value is provided in the displaytext part of a Hyperlink field, the value displayed by Microsoft Access will be the same as the displaytext setting. When there's no value in the *displaytext* part of a Hyperlink field, Microsoft Access displays the value of the *address* or *subaddress* part of the Hyperlink field, depending on which value is first present in the field.

The following table shows the values returned by the **HyperlinkPart** function for data stored in a Hyperlink field.

Hyperlink field data	HyperlinkPart function returned values
#http://www.microsoft.com/#	**acDisplayedValue**: http://www.microsoft.com/ **acDisplayText**: No value returned. **acAddress**: http://www.microsoft.com/ **acSubAddress**: No value returned.
Microsoft#http://www.microsoft.com/#	**acDisplayedValue**: Microsoft **acDisplayText**: Microsoft **acAddress**: http://www.microsoft.com/ **acSubAddress**: No value returned.
Customers##Form Customers	**acDisplayedValue**: Customers **acDisplayText**: Customers **acAddress**: No value returned. **acSubAddress**: Form Customers
##Form Customers	**acDisplayedValue**: Form Customers **acDisplayText**: No value returned.

acAddress: No value returned.
acSubAddress: Form Customers

The following example uses all four of the part argument constants to display information returned by the **HyperlinkPart** function for each record in a table containing a Hyperlink field. To try this example, paste the DisplayHyperlinkParts procedure into the Declarations section of a module. You can call the DisplayHyperlinkParts procedure from the Debug window, passing to it the name of a table that contains hyperlinks and the name of the field that contains Hyperlink data, as shown in the following example.

```
DisplayHyperlinkParts "MyHyperlinkTableName", "MyHyperlinkFieldName"

Sub DisplayHyperlinkParts(strTable As String, strField As String)
    Dim dbs As Database, rst As Recordset
    Dim strMsg As String

    Set dbs = CurrentDb
    Set rst = dbs.OpenRecordset(strTable)

    While Not rst.EOF       ' For each record in table.
        strMsg = "DisplayValue = " & HyperlinkPart(rst(strField),
acDisplayedValue) _
            & vbCrLf & "DisplayText = " & HyperlinkPart(rst(strField),
acDisplayText) _
            & vbCrLf & "Address = " & HyperlinkPart(rst(strField), acAddress) _
            & vbCrLf & "SubAddress = " & HyperlinkPart(rst(strField),
acSubAddress)
        ' Show parts returned by HyperlinkPart function.
        MsgBox strMsg
        rst.MoveNext
    Wend
End Sub
```

When you use the **HyperlinkPart** function in a query, the part argument is required. For example, the following SQL statement uses the **HyperlinkPart** function to return information about data stored as a Hyperlink data type in the URL field of the Links table:

```
SELECT Links.URL, HyperlinkPart([URL],0)
AS Display, HyperlinkPart([URL],1)
AS Name, HyperlinkPart([URL],2)
AS Addr, HyperlinkPart([URL],3) AS SubAddr
FROM Links;
```

For another example of using the **HyperlinkPart** function, see Displaying a Document in the WebBrowser Control by Using a Hyperlink Stored in a Table later in this chapter.

When you use the **Follow** method in Microsoft Access, you don't need to know the address specified by a control's **HyperlinkAddress** or **HyperlinkSubAddress** property, or by the Hyperlink field that is bound to a text box, list box, or combo box control. You only need to know the name of the control that contains the hyperlink.

This example uses the **Follow** method to automatically open the Web page specified in a text box bound to a Hyperlink field on a form whenever the user moves to a new record. Add the following code to the OnCurrent event of a form.

```
Private Sub Form_Current()
    Dim txt As TextBox

    On Error GoTo Error_Form1

    ' Set reference to the txtAddress text box bound to a Hyperlink field.
    Set txt = txtAddress

    ' Follow the hyperlink.
    txt.Hyperlink.Follow

Exit_Form1:
    Exit Sub

Error_Form1:
    MsgBox Err & ": " & Err.Description
    Resume Exit_Form1
End Sub
```

For another example of following a hyperlink stored in a table, see Displaying a Document in the WebBrowser Control by Using a Hyperlink Stored in a Table later in this chapter.

You can use Data Access Objects (DAO) code to create a field with the Hyperlink field type. To do so, you must first create a field with the Memo data type and then set the **Attributes** property of the field to **dbHyperlinkField**. The following example creates a table named Hyperlinks that contains a Text field and a Hyperlink field.

```
Sub CreateHyperlinkField()
    Dim db As Database
    Dim tbl As TableDef

    Set db = CurrentDb()

    Set tbl = db.CreateTableDef("Hyperlinks")

    With tbl
        .Fields.Append .CreateField("Text", dbText)
        .Fields.Append .CreateField("Hyperlink", dbMemo)
        .Fields("Hyperlink").Attributes = dbHyperlinkField
    End With

    db.TableDefs.Append tbl
    RefreshDatabaseWindow
End Sub
```

All of the Office 97 applications provide ways to save their data as HTML documents. Microsoft Access and Word provide ways of doing so by using Visual Basic.

This section includes the following topics:

Microsoft Access has five ways to save data from your database as HTML documents:

- **Save data as static HTML documents**

  You can create *static* HTML documents from table, query, and form datasheets, and from reports. When you save data as static HTML documents, the resulting pages reflect the state of the data at the time it was saved, like a snapshot. If your data changes, you must save the pages again to share the new data.

- **Save table, query, and form datasheets as IDC/HTX files**

  You can save your table, query, and form datasheets as Internet Database Connector/HTML extension (IDC/HTX) files that generate HTML documents by querying a copy of your database located on a Web server for current data.

- **Save forms and datasheets as Active Server Pages**

  You can save your forms as Active Server Pages (ASP) that emulate most of the functionality of your forms and display data from a database located on a Web server. You can also save table, query, and form datasheets as Active Server Pages that display current data from a copy of your database located on a Web server.

- **Automate the publishing of dynamic and static HTML documents by using the Publish to the Web Wizard**

  You can use the Publish to the Web Wizard to automate the process of saving multiple objects to any combination of all three file types. In the Publish to the Web Wizard, IDC/HTX files and Active Server Pages (ASP) files are collectively referred to as *dynamic* Web pages because these file types create HTML documents by querying the database to include current data.

- **Automate the publishing of dynamic and static HTML documents by using the OutputTo method or action**

  You can use the **OutputTo** method in Visual Basic and the OutputTo action in macros to automate the process of saving objects to any of the three file types.

The following sections discuss each of these options in more detail.

This section includes the following topics:

With Microsoft Access, you can save table, query, and form datasheets, and reports as static HTML documents.

To save a table, query, or form datasheet, or a report as a static HTML document

1. In the Database window, click the table, query, form, or report you want to save.
2. On the **File** menu, click **Save As/Export**.
3. In the **Save As** dialog box, click **To an External File or Database**, and then click **OK**.
4. In the **Save as type** box, click **HTML Documents (*.html; *.htm)**.
5. If you want to preserve formatting, select the **Save Formatted** check box. To automatically open the resulting HTML document in your Web browser, select the **Autostart** check box.
6. Specify the file name and location to save the file, and then click **Export**.
7. In the **HTML Output Options** dialog box, if you want Microsoft Access to merge an HTML template with the resulting HTML document, specify that as well, and then click **OK**.

For information about HTML templates, see “Using an HTML Template When You Save Data as HTML Documents” later in this section.

You can also save data as static HTML documents by using the Publish to the Web Wizard (available through the **Save As HTML** command on the **File** menu), the **OutputTo method** in code, or the OutputTo action in macros. For information about using the **OutputTo** method, see Saving HTML Documents by Using the OutputTo Method later in this chapter.

When saving table, query, and form datasheets, Microsoft Access saves each datasheet to a single HTML file. Microsoft Access saves reports as multiple HTML documents, with one HTML file per printed page. To name each page, Microsoft Access uses the name of the object and appends _Page*nn* to the end of each page’s file name after the first page; for example, ProductList.htm, ProductList_Page2.htm, ProductList_Page3.htm, and so on.

Saving Table, Query, and Form Datasheets as Static HTML Documents

When you save a table, query, or form datasheet as an HTML document, the HTML document generated is based on the table or query associated with the datasheet, including the current setting of the **OrderBy** or **Filter** property of the table or query. If the datasheet contains a parameter query, Microsoft Access first prompts you for the parameter values, then exports the results.

If you select the **Save Formatted** check box, the HTML document contains an HTML table that reflects as closely as possible the appearance of the datasheet by using the appropriate HTML tags to specify color, font, and alignment. The HTML document follows as closely as possible the page orientation and margins of the datasheet. Whenever you want to use settings that are different from the default orientation and margins for a datasheet, you must first open the datasheet, and then use the **Page Setup** command (**File** menu) to change settings before you save the datasheet as an HTML document.

If you select the **Save Formatted** check box, and a field has a **Format** or **InputMask** property setting, those settings are reflected in the data in the HTML document. For example, if a field’s **Format** property is set to Currency, the data in the HTML document is formatted with a dollar sign, a comma as the thousand separator, and two decimal places; for example, $1,123.45.

Saving Reports as Static HTML Documents

When you save a report as HTML documents, the series of HTML documents generated is based on the report’s underlying table or query, including the current **OrderBy** or **Filter** property settings of the table or query. If the report contains a parameter query, Microsoft Access first prompts you for the parameter values, then exports the results.

The HTML documents simulate as closely as possible the appearance of the report by creating the appropriate HTML tags to retain attributes such as color, font, and alignment. The proportions and layout of the actual report follow as closely as possible the page orientation and margins set for the report. To change the page orientation and margins, open the report in Print Preview, and then use the **Page Setup** command to change settings before you save the report as HTML documents. These settings are saved from session to session for reports,

so if you change them once, they will be used the next time you save the form or report as HTML documents.

Most controls and features of a report, including subreports, are supported except the following: lines, rectangles, OLE objects, and subforms. However, you can use an HTML template file to include report header and footer images in your output files. For an example, see the Nwindtem.htm template file in the C:\Program Files\Microsoft Office\Office\Samples folder.

Navigation Controls When Saving Multiple HTML Documents Per Object

If you specify an HTML template that contains placeholders for navigation controls when you save a report as multiple HTML documents, Microsoft Access creates hyperlinks that the user can use to go to the first, previous, next, and last pages in the publication. Where Microsoft Access places the hyperlinks depends on where you locate the placeholders in the HTML template. For information about HTML templates and placeholders, see "Using an HTML Template When You Save Data as HTML Documents" later in this section.

How Microsoft Access Saves Data Types in HTML Format

When you save data as static HTML documents, Microsoft Access saves values from most data types as strings and formats them as closely as possible to their appearance in the datasheet or report. All unformatted data types, except Text and Memo, are saved with right alignment as the default. Text and Memo fields are saved with left alignment by default.

There are two exceptions:

® OLE Object fields are not saved.

® Hyperlink field values are saved as hyperlinks in the HTML document. The hyperlinks use HTML anchor tags with an HREF attribute, as described in the following table.

If	**Anchor tag format**
The hyperlink doesn't include a subaddress	<A HREF="*address*">*displaytext*</A>
The hyperlink includes a subaddress	<A HREF="*address#subaddress*">*displaytext*</A>
Display text isn't specified	<A HREF="*address*">*address*</A>

Microsoft Access determines the *displaytext*, *address*, and *subaddress* values by parsing the value stored in the Hyperlink field. For information about the *displaytext*, *address*, and *subaddress* values, see The Hyperlink Field Storage Format earlier in this chapter.

Using an HTML Template When You Save Data as HTML Documents

When you save data as HTML documents, you can use an HTML template to give a consistent look to the HTML documents you create. For example, you can include your company's logo, name, and address in the page's header, use the background that is used throughout your company, or include standard text in the header or footer of the HTML document.

Note You can use an HTML template when you save data as static HTML documents, when you save datasheets as IDC/HTX files, when you save a form or datasheet as an Active Server Page, and when you use the Publish to the Web Wizard.

The HTML template can be any HTML document; that is, a text file that includes HTML tags and user-specified text and references. In addition, the HTML template can include placeholders that tell Microsoft Access where to insert certain pieces of data in the HTML documents. When data is saved as HTML documents, the placeholders are replaced with data. The following table describes each of the placeholders that you can use in

an HTML template.

Placeholder	Description	Location
<!--AccessTemplate_Title-->	The name of the object being saved	Between <TITLE> and </TITLE>
<!--AccessTemplate_Body-->	The data or object being saved	Between <BODY> and </BODY>
<!--AccessTemplate_FirstPage-->	An anchor tag to the first page	Between <BODY> and </BODY> or after </BODY>
<!--AccessTemplate_PreviousPage-->	An anchor tag to the previous page	Between <BODY> and </BODY> or after </BODY>
<!--AccessTemplate_NextPage-->	An anchor tag to the next page	Between <BODY> and </BODY> or after </BODY>
<!--AccessTemplate_LastPage-->	An anchor tag to the last page	Between <BODY> and </BODY> or after </BODY>
<!--AccessTemplate_PageNumber-->	The current page number	Between <BODY> and </BODY> or after </BODY>

When you install Microsoft Access, sample HTML template files and graphics files are installed in the Access subfolder of the Templates folder. The default location of this folder is C:\Program Files\Microsoft Office\Templates\Access.

With Microsoft Access, you can save a table, query, or form datasheet as Internet Database Connector/HTML extension (IDC/HTX) files that generate HTML documents by querying a copy of your database located on a Web server. In contrast to static HTML documents, which contain the data that was current at the time the HTML document was created, IDC/HTX files generate an HTML page with current data from your database; therefore, the HTML documents that they generate are called dynamic.

u **To save a table, query, or form datasheet as IDC/HTX files**

1. In the Database window, click the table, query, or form you want to save.
2. On the **File** menu, click **Save As/Export**.
3. In the **Save As** dialog box, click **To an External File or Database**, and then click **OK**.
4. In the **Save as type** box, click **Microsoft IIS 1-2 (*.htx;*.idc)**.
5. Specify the file name and location to save the files, and then click **Export**.
6. In the **HTX/IDC Output Options** dialog box, specify:
   - ® The data source name that will be used for the database.
   - ® A user name and password, if required to open the database.
   - ® An HTML template, if you want Microsoft Access to merge one with the HTML extension (HTX) file.

   Note You can specify any of these items later, except the HTML template, by editing the resulting IDC file in a text editor such as Notepad.
7. Click **OK**.

If the datasheet contains a parameter query, Microsoft Access simulates **the Enter Parameter Value** dialog box by creating an additional HTML parameter page that contains an HTML form text box control to enter the parameter value and a button to run the query. You must display this HTML parameter page before you display the datasheet HTML page in your Web application. If you use the Publish to the Web Wizard and you specify a switchboard page, the HTML parameter page is added to the switchboard page. When you export, Microsoft Access runs the query and displays the **Enter Parameter Value** dialog box. You don't need to enter values in this dialog box — just click **OK** to continue.

You can also save a table, query, or form datasheet as IDC/HTX files by using the Publish to the Web Wizard (available through the **Save As HTML** command on the **File** menu), the **OutputTo** method in code, or the OutputTo action in macros. For information about using the **OutputTo** method, see Saving HTML Documents by Using the OutputTo Method later in this chapter.

How the Internet Database Connector Works

When you save a table, form, or query datasheet as Internet Connector files, Microsoft Access creates two files: an *Internet Database Connector (IDC) file* and *HTML extension (HTX) file*. These files are used to generate a Web page that displays current data from your database.

An IDC file contains the necessary information to connect to a specified Open Database Connectivity (ODBC) data source and to run an SQL statement that queries the database. The information needed to connect to the database includes the data source name, and if user-level security is established for the database, the user name and password required to open the database. For example, if you save the Current Product List query datasheet from the Northwind sample application as IDC/HTX files, Microsoft Access creates the following IDC file:

```
Datasource:Northwind
Template:Current Product List.htx
SQLStatement:SELECT [Product List].ProductID, [Product List].ProductName
+FROM Products AS [Product List]
+WHERE ((([Product List].Discontinued)=No))
+ORDER BY [Product List].ProductName;

Password:
Username:
```

An IDC file also contains the name and location of an HTML extension (HTX) file. The HTX file is a template for the HTML document; it contains field merge codes that indicate where the values returned by the SQL statement should be inserted. For example, if you save the Current Product List query datasheet from the Northwind sample application as IDC/HTX files, Microsoft Access creates the following HTX file:

```
<HTML>
<TITLE>Current Product List</TITLE>
<BODY>
<TABLE BORDER=1 BGCOLOR=#ffffff><FONT FACE="Arial" COLOR=#000000>
<CAPTION><B>Current Product List</B></CAPTION>
<THEAD>
<TR>
<TD><FONT SIZE=2 FACE="Arial" COLOR=#000000>Product ID</FONT></TD>
<TD><FONT SIZE=2 FACE="Arial" COLOR=#000000>Product Name</FONT></TD>
</TR>
</THEAD>
<TBODY>
<%BeginDetail%>
<TR VALIGN=TOP>
<TD ALIGN=RIGHT><FONT SIZE=2 FACE="Arial"
COLOR=#000000><%ProductID%><BR></FONT></TD>
<TD><FONT SIZE=2 FACE="Arial" COLOR=#000000><%ProductName%><BR></FONT></TD>
</TR>
<%EndDetail%>
</TBODY>
<TFOOT></TFOOT>
</BODY>
</HTML>
```

Microsoft Access saves the HTX file to be used with an IDC file with the same name as the IDC file, except with an .htx file name extension rather than an .idc file name extension. After the database information has been merged into the HTML document, it is returned to the Web browser.

If you open Current Product List.idc from a Microsoft Internet Information Server that has an appropriately defined Northwind data source name (DSN), the Web page shown in the following illustration is generated.

{ewc mvimg, mvimage,!illust.bmp}

Note You can also reference an HTML template when you create IDC and HTX files. An HTML template contains additional HTML code to enhance the appearance of the resulting pages. If you specify an HTML template, it is merged with the HTX file. For information about the format of an HTML template, see “Using an HTML Template When You Save Data as HTML Documents” in Saving Data as Static HTML Documents, earlier in this chapter.

Requirements for Using IDC/HTX Files

To use IDC/HTX files, your database and the IDC/HTX files must reside on a computer running one of the following operating systems and Internet server platforms:

® Microsoft Windows NT Server version 3.51 or 4.0 running Microsoft Internet Information Server version 1.0, 2.0, or 3.0

® Microsoft Windows NT Workstation version 4.0 and Microsoft Peer Web Services

® Microsoft Windows 95 and Microsoft Personal Web Server

Microsoft Internet Information Server, Microsoft Peer Web Services, and Microsoft Personal Web Server use a component called the Internet Database Connector (Httpodbc.dll) to generate dynamic Web pages from IDC/HTX files.

The Internet Database Connector component requires ODBC drivers to access a database. To access a Microsoft Access database, the Microsoft Access Desktop driver (Odbcjt32.dll) must be installed on your Web server. This driver is installed when you install Microsoft Internet Information Server if you select the **ODBC Drivers And Administration** check box during Setup.

However, the Microsoft Access Desktop driver isn't installed with Microsoft Personal Web Server. If Microsoft Access is installed on the computer you are using to run Microsoft Personal Web Server, and if you selected the driver when you installed Microsoft Access, the driver is already available. If you don't have Microsoft Access installed on the computer you are using to run Microsoft Personal Web Server, you must install the Microsoft Access Desktop driver.

u **To install the Microsoft Access Desktop driver**

1. Run the Microsoft Office or Microsoft Access Setup program.
2. If you are running Setup for the first time, click **Custom**.

   If you are not running Setup for the first time, click **Add/Remove**.
3. Select the **Data Access Controls** check box, and then click **Change Option**.

Important The **Microsoft Access** check box must also be selected or the driver will not be installed.

---

4. Select the **Database Drivers** check box, and then click **Change Option**.
5. Select the **Microsoft Access Driver** check box, and then click **OK**.
6. Click **Continue**, and follow the instructions in the remaining Setup dialog boxes.

After the Microsoft Access Desktop driver is installed, you must create either a system DSN or a file DSN that specifies the name and connection information for each database you want to use on the server. You then specify that DSN when you generate the IDC/HTX files.

For information about how to define a system DSN or a file DSN, search the Microsoft Access Help index for "ODBC, setting up data sources." For more information about using IDC/HTX files, search the Microsoft Internet Information Server Help index for "database connector.

Click here for more information about Microsoft Internet Information Server.
{ewc mvimg, mvimage,!intjump.bmp}

{ewc mvimg, mvimage,!tip.bmp}

With Microsoft Access, you can save a form as an Active Server Page (ASP) that emulates much of the functionality of your form. When saving a form as an Active Server Page, Microsoft Access saves most, but not all, controls on the form as ActiveX controls that perform the same or similar functions. Microsoft Access doesn't save or run Visual Basic code behind the form or controls. To copy the layout of your form as closely as possible, Microsoft Access uses the Microsoft HTML Layout control to position the controls on Active Server Pages. The resulting page uses ActiveX Scripting and ActiveX Data Objects to connect the control on the page to a copy of your database on an Internet server.

Click here for information about the Microsoft HTML Layout control.
{ewc mvimg, mvimage,!intjump.bmp}

Users who open a form saved as an Active Server Page can browse records, update or delete existing records, and add new records by using a Web browser.

You can also save table, query, and form datasheets as Active Server Pages. When you open a datasheet saved as an ASP, Microsoft Access displays current data from a copy of your database located on an Internet server, much like IDC/HTX files do. However, unlike IDC/HTX files, Active Server Pages require only one file per datasheet. The ASP file uses scripting to establish a connection to the database on the server, and contains information that it uses to format the datasheet. Unlike a form saved as an Active Server Page, users can't update existing records in or add new records to a datasheet saved as an Active Server Page.

u **To save a form or datasheet as an Active Server Page**

1. In the Database window, click the form or datasheet you want to save.
2. On the **File** menu, click **Save As/Export**.
3. In the **Save As** dialog box, click **To an External File or Database**, and then click **OK**.
4. In the **Save as type** box, click **Microsoft Active Server Page (*.asp)**.
5. Specify the file name and location to save the file, and then click **Export**.
6. In the **Active Server Page Output Options** dialog box, specify:
   - ® The data source name that will be used for a copy of the current database (required).
   - ® A user name and password, if required to open the database.
   - ® An HTML template, if you want Microsoft Access to merge one with the Active Server Page.

     For information about HTML templates, see "Using an HTML Template When You Save Data as HTML Documents" in Saving Data as Static HTML Documents, earlier in this chapter.
   - ® The URL for the server where the Active Server Page will reside.
   - ® The **Session timeout** setting, which determines how long a connection to the server is maintained after the user stops working with the Active Server Page (optional).
7. Click **OK**.

You can also save forms and datasheets as Active Server Pages by using the Publish to the Web Wizard (available through the **Save As HTML** command on the **File** menu), the **OutputTo** method in code, or the OutputTo action in macros. For information about using the **OutputTo** method, see Saving HTML Documents by Using the OutputTo Method later in this chapter.

Form Views Supported for Active Server Pages

If the form you save as an Active Server Page has its **DefaultView** property set to Single Form or Continuous Forms, the Active Server Page displays as a single form, unless it is open in Datasheet view when you use the **Save As/Export** command (**File** menu). If the form has its **DefaultView** property set to Datasheet, the Active Server Page displays as a datasheet. Subforms always display as datasheets, regardless of their **DefaultView** property setting. All field data types are saved unformatted, that is, **Format** and **InputMask** property settings aren't saved.

Control Types Supported for Active Server Pages

When Microsoft Access saves a form as an Active Server Page, it replaces Microsoft Access controls with ActiveX controls, as described in the following table.

Microsoft Access control	ActiveX control
Text box	Text box.
Text box control bound to a Hyperlink field	Text box that displays the hyperlink text, but the hyperlink can't be followed.
List box	List box.
Combo box	Combo box.
Label	Label. If the label has **HyperlinkAddress** and/or **HyperlinkSubAddress** properties set, an HTML hyperlink is created for the label.
Command button	Command button, but any code behind the button isn't saved. If the command button has **HyperlinkAddress** and/or **HyperlinkSubAddress** properties set, an HTML hyperlink is created for the button.
Option group	Option group, but without a group frame.
Option button	Option button.
Check box	Check box.
Toggle button	Toggle button.
ActiveX control	ActiveX control, but any code behind the control isn't saved.
Subform	Subform as datasheet only.

Microsoft Access doesn't support the following controls when saving a form as an Active Server Page:

® Tab control, and anything on a tab control

® Rectangle

® Line

® Page break

® Unbound object frame

® Bound object frame

® Image control

® The background of a form set with the Picture property

Note You can simulate a rectangle or a line by using a Label control without a caption.

Requirements for Using Active Server Pages

To display and use an Active Server Page, a copy of your database and Active Server Pages must reside on a computer running one of the following operating systems and Internet server platforms:

® Microsoft Windows NT Server version 3.51 or 4.0 running Microsoft Internet Information Server version 3.0 (which includes Active Server Pages components)

® Microsoft Windows NT Workstation version 4.0 and Microsoft Peer Web Services with the Active Server Pages components installed

® Microsoft Windows 95 and Microsoft Personal Web Server with the Active Server Pages components installed

The Microsoft HTML Layout control must be installed on the computer opening the Active Server Page. If the Microsoft HTML Layout control isn't installed on a computer opening a Microsoft Access Active Server Page, the user will be prompted to download this control from the Microsoft Web site.

If you are already using Microsoft Windows NT Server version 3.51 or 4.0, but don't have Microsoft Internet Information Server version 3.0, you can read about how to download and install it by connecting to http://www.microsoft.com/iis/. Before you can use Active Server Pages with Microsoft Peer Web Services or

Microsoft Personal Web Server, you must download and install the Active Server Pages components after installing your personal web server software. To do this, connect to http://www.microsoft.com/iis/, choose to download IIS 3.0, register to download, and select to download the Active Server Pages components. Active Server Pages also require the Microsoft Access Desktop driver and a valid DSN to access a database. For information about installing the Microsoft Access Desktop driver and defining DSNs, see “Requirements for Using IDC/HTX Files” in Saving Table, Query, and Form Datasheets as IDX/HTX Files, earlier in this chapter.

You can use the **OutputTo method** to save Microsoft Access database objects in the HTML formats described in the previous sections: static HTML documents, IDC/HTX files, or Active Server Pages (ASP).

The syntax of the **OutputTo** method is:

DoCmd.OutputTo *objecttype*, *objectname*, *outputformat*, *outputfile*, *autostart*, *templatefile*

The following table describes the arguments of the OutputTo method.

Argument	Description
objecttype	Required. Specifies the type of database object you are going to output. You can use one of the following constants for the objecttype argument: **acOutputForm** **acOutputQuery** **acOutputReport** **acOutputTable**
objectname	Optional. A string expression that's the valid name of an object of the type specified in the objecttype argument. If you want to output the active object, specify the object's type for the objecttype argument and leave this argument blank. If you run Visual Basic code that contains the **OutputTo** method in a library database, Microsoft Access looks for the object with this name first in the library database, then in the current database.
outputformat	Optional. Specifies whether to save the database object as an HTML document, IDC/HTX file, or Active Server Page. You can use one of the following constants for the outputformat argument: **acFormatHTML** **acFormatIIS** **acFormatASP** If you leave this argument blank, Microsoft Access prompts you for the output format.
outputfile	Optional. A string expression that's the full name, including the path, of the file you want to output the object to. You can include the standard file name extension (.asp, .htm, .html, or .htx,) for the output format you select with the outputformat argument, but it's not required. If you output to IDC/HTX or ASP files, Microsoft Access always creates files with the standard .htx and .idc or .asp file name extensions. If you leave this argument blank, Microsoft Access prompts you for an output file name.
autostart	Optional. Use **True** (-1) to start a Web browser immediately to open the static HTML document specified by the outputfile argument. Use **False** (0) if you don't want to start the application. This argument is ignored for IDC/HTX and ASP files. If you leave this argument blank, Microsoft Access uses the default value (**False**).
templatefile	Optional. A string expression that's the full name, including the path, of the file you want to use as a template for an HTML, IDC/HTX, or ASP file.

Microsoft Internet Information Server and Microsoft Active Server Pages formats are available only for tables, queries, and forms, so if you specify **acFormatIIS** or **acFormatASP** for the *outputformat* argument, you must specify **acOutputTable**, **acOutputQuery**, or **acOutputForm** for the *objecttype* argument.

You can leave an optional argument blank in the middle of the syntax, but you must include the argument's comma. If you leave a trailing argument blank, don't use a comma following the last argument you specify.

You can’t specify the HTML template, data source name, user name and password, server URL, or **Session timeout** setting when you use the **OutputTo** method. Microsoft Access uses the values specified on the **Hyperlinks/HTML** tab of the **Options** dialog box (**Tools** menu) by default. However, you can use the **SetOption** method in your code to temporarily change these settings. For information about using the **SetOption** method, search the Microsoft Access Help index for “SetOption method.

Examples

The following example outputs the Employees table in static HTML document format to the Employee.htm file and immediately opens the file in the default Web browser.

```
DoCmd.OutputTo acOutputTable, "Employees", acFormatHTML, "Employee.htm", True
```

The following example outputs the Employees table in IDC/HTX format to two files named Employee.htx and Employee.idc. It merges the Mc.htm template file into the Employee.htx file.

```
DoCmd.OutputTo acOutputTable, "Employees", acFormatIIS, "Employee",, _
    "C:\Program Files\Microsoft Office\Templates\Access\Mc.htm"
```

The following example outputs the Products form in Active Server Page format to the Products.asp file. It merges the Stones.htm template file into the Products.asp file.

```
DoCmd.OutputTo acOutputFor
321m, "Products", acFormatASP, "Products",, _
    "C:\Program Files\Microsoft Office\Templates\Access\Stones.htm"
```

You can save an existing Word document to HTML format by using the Save As command (File menu) or by using Visual Basic code. The following example saves the active document as an HTML document.

```
Sub SaveAsHTML
    Dim intFormat As Integer

    intFormat = FileConverters("HTML").SaveFormat
    myDocName = ActiveDocument.Name
    pos = InStr(myDocName, ".")
    If pos > 0 Then
        myDocName = Left(myDocName, pos -1)
        myDocName = myDocName & ".html"
        ActiveDocument.SaveAs FileName:=myDocName, FileFormat:=intFormat
    End If
End Sub
```

When you save an existing Word document to HTML format, formatting and other items that aren't supported by HTML or the Word Web page authoring environment are removed from the file. For more information about what happens when you save a Word document as a Web page, search Word Help. Instead of saving existing documents as HTML, you may want to create new HTML documents with Microsoft Word Web authoring tools.

Word 97 has many powerful features for creating HTML documents, such as the following:

® **Word Editing and Formatting Features**

Take advantage of advanced Word editing and formatting features — rich-text formatting, spelling and grammar checking, and automatic text correction — when you work with Word Web authoring tools. When you use a Word Web template, you can easily create and format popular Web page items — such as tables, bulleted or numbered lists, and graphic objects — just as you can with a regular Word document.

® **Word Web Templates**

Use the Web Page Wizard or the Blank Web Page template to create new Web pages. The Web Page Wizard gives you different layouts and color themes to choose from, such as a personal home page, a table of contents, a survey, or a registration form. To use the wizard or the template, click **New** on the **File** menu, click the **Web Pages** tab, and then double-click **Web Page Wizard** or **Blank Web Page**. There are several additional Web templates that you can download from the Microsoft Word Web site at http://www.microsoft.com/word/. When you download these templates, they are installed in the same folder as the existing Web templates.

Click here to connect to the Microsoft Word Web site.
{ewc mvimg, mvimage,!intjump.bmp}

® **Hyperlinks, Bullets, and Horizontal Lines**

By using the **Insert Hyperlink** button on the **Standard** toolbar, you can quickly create hyperlinks on your Web page to link related information in different locations. The hyperlink text is usually blue and underlined. You can also quickly create special graphical bulleted lists and horizontal lines for your Web page. To add a new bullet for selected text, click **Bullets and Numbering** on the **Format** menu, and then select the bullet you want. To add the default bullet to selected text, click **Bullets** on the **Formatting** toolbar. To add a new horizontal line, click **Horizontal Line** on the **Insert** menu, and then select the line style you want.

® **Forms**

You can use forms to collect and present data on your Web page. For example, you can publish a form that collects user feedback or registration information. You can store the input data in a database or a text file for future use. You can quickly create a form by selecting a sample form and then modifying it for your needs by using the Forms toolbar. To select a sample form, click **New** on the **File** menu, click the **Web Pages** tab, and then select the sample form you want. To display the **Control Toolbox**, click **Form Design Mode** on the **Standard** toolbar. Use the Web form tools just as you use the regular Word form tools to insert form elements.

To make your Microsoft Excel data available to users on your intranet or the World Wide Web, use the Internet Assistant add-in program to convert worksheet data or charts to HTML Web pages (**Save As HTML** command, **File** menu). Microsoft Excel doesn't support using Visual Basic to save data as HTML documents.

To make your Microsoft PowerPoint data available to users on your intranet or the World Wide Web, use the Internet Assistant add-in program to convert presentations to HTML Web pages (**Save As HTML** command, **File** menu). Microsoft PowerPoint doesn't support using Visual Basic to save data as HTML documents.

Each Office application provides features that you can use to open and import HTML data. You can also use Office applications to open documents and files in a variety formats on your company's intranet. If you have a connection to the Internet, you can open or import data in most of these same formats on Internet sites such as FTP and HTTP servers.

Note In all Office applications except Outlook, you can use Data Access Objects (DAO) code in Visual Basic to access and manipulate data in a variety of formats, including HTML. For more information about using DAO, see Chapter 11, "Data Access Objects.

Opening HTML Data in Microsoft Word

To open HTML documents in Microsoft Word with Visual Basic, use the **Open** method. By default, the **Open** method tries each available file converter until it succeeds. For this reason, as long as the HTML Document converter is installed, the following example opens an HTML document on a local drive.

```
Documents.Open "C:\My Documents\My Document.htm"
```

Similarly, you can specify a URL to open a file located on an HTTP server, as follows.

```
Documents.Open "http://myserver.com/default.htm"
```

To improve performance, you can specify the file converter to use by referring to it in the **FileConverters** collection, as follows.

```
Sub OpenHTML()
    Dim intFormat As Integer

    intFormat = FileConverters("HTML").OpenFormat
    Documents.Open "http://myserver.com/default.htm", Format:=intFormat
End Sub
```

Opening HTML Data in Microsoft Excel

To open HTML documents in Microsoft Excel with Visual Basic, use the **Open** method. You don't need to specify the file converter to use, because the **Open** method tries each available file converter until it succeeds. The following example opens an HTML document on a local drive.

```
Workbooks.Open "C:\My Documents\Product List.htm"
```

Similarly, you can specify a URL to open a file located on an HTTP server, as follows.

```
Workbooks.Open "http://m;yserver.com/default.htm"
```

You can also get data from an intranet site or from HTTP, FTP, or Gopher sites on the World Wide Web by running a Web query. To run a Web query, point to **Get External Data** on the **Data** menu, and then click **Run Web Query**.

For more information about running Web queries, click here to connect to the Excel Webquery Web page.
{ewc mvimg, mvimage,!intjump.bmp}

Opening HTML Data in Microsoft PowerPoint

To open HTML documents in PowerPoint with Visual Basic, use the Open method. By default, the **Open** method tries each available file converter until it succeeds. For this reason, as long as the HTML Document converter is installed, the following line of code will open an HTML document on a local drive:

```
Presentations.Open "C:\My Documents\My Document.htm"
```

Similarly, you can specify a URL to open a file located on an HTTP server:

```
Presentations.Open "http://myserver.com/default.htm"
```

Importing HTML Data in Microsoft Access

With Microsoft Access, you can import or link data from HTML tables or other data sources on an Internet server. For more information about importing, exporting, and linking HTML data and other data formats on Internet servers, see “Working with HTML Files” in Chapter 18 and “Importing, Linking, and Exporting Data on the Internet” in Chapter 21 in Building Applications with Microsoft Access 97.

The Microsoft WebBrowser control is an ActiveX control that you can use to browse Web sites, view Web pages and other documents, and download data located on the Internet from your applications. The WebBrowser control is useful in situations where you don't want to disrupt the work flow in your application by switching to a Web browser or other document-viewing application.

The WebBrowser control can display any Web page that Microsoft Internet Explorer version 3.0 can display. For example, the WebBrowser control can display pages that include any of the following features:

® Standard HTML and most HTML enhancements, such as floating frames and cascading style sheets

® Other ActiveX controls

® Most Netscape plug-ins

® Scripting, such as Microsoft Visual Basic Scripting Edition (VBScript) or JavaScript

® Java applets

® Multimedia content, such as video and audio playback

® Three-dimensional virtual worlds created with Virtual Reality Modeling Language (VRML)

With the WebBrowser control, users of your application can browse sites on the World Wide Web, as well as folders on a local hard disk and on a local area network. Users can follow hyperlinks by clicking them or by typing a URL into a text box. Also, the WebBrowser control maintains a history list that users can browse through to view previously browsed sites, folders, and documents.

In addition to opening Web pages, both Microsoft Internet Explorer version 3.0 and the WebBrowser control can open any ActiveX document, which includes most Office documents. For example, if Office is installed on a user's computer, an application that uses the WebBrowser control can open and edit Microsoft Excel workbooks, Word documents, and PowerPoint presentations from within the control. Similarly, if Microsoft Excel Viewer, Word Viewer, or PowerPoint Viewer is installed, users can open those documents within the WebBrowser control, but they cannot edit them.

You can't open and edit a Microsoft Access database as an ActiveX document within Microsoft Internet Explorer version 3.0 or the WebBrowser control, but a Web page can contain a hyperlink to a Microsoft Access database. Clicking the hyperlink downloads a copy of the database and starts a session of Microsoft Access to open it. Additionally, if you have the server software that supports Internet Database Connector/HTML extension (IDC/HTX) files or Active Server Pages (ASP), you can create Web pages that act as a front-end to an ODBC data source such as a Microsoft Access or Microsoft SQL Server database. For more information about creating IDC/HTX or ASP files, see Saving Microsoft Access Data as HTML Documents earlier in this chapter.

This sections includes the following topics:

Before you can use the WebBrowser control, you must have Microsoft Internet Explorer version 3.0 installed.

If you purchased Microsoft Office 97 on CD-ROM, you can install Microsoft Internet Explorer version 3.0 by running Msie30.exe from the Iexplore subfolder in the ValuPack folder.

If you prefer to install from the Web, you can download and install Microsoft Internet Explorer version 3.0 from http://www.microsoft.com/ie/download/.

Once you have Microsoft Internet Explorer version 3.0 installed, the WebBrowser control is automatically registered and is available in form Design view (Microsoft Access) and in Design mode (Microsoft Excel, Word, and PowerPoint).

u **To add the WebBrowser control to a document or form**

1. In Microsoft Excel, Word, or PowerPoint, open the document or form. In Microsoft Access, open the form in Design view.
2. In Microsoft Excel, Word, or PowerPoint, right-click the menu bar and then click **Control Toolbox**. In Microsoft Access, right-click the menu bar and then click **Toolbox**.
3. In the toolbox, click the **More Controls** tool.

   A menu appears that lists all the registered ActiveX controls in your system.
4. On the menu of ActiveX controls, click Microsoft WebBrowser Control.
5. On the document or form, click where you want to place the control.
6. Move and size the control to the area you want to display.

In Microsoft Excel, Word, and PowerPoint, you can also add the WebBrowser control to UserForms created with the Visual Basic editor.

u **To add the WebBrowser control to a UserForm created with the Visual Basic Editor**

1. Open a Microsoft Excel, Word, or PowerPoint document.
2. On the **Tools** menu, point to **Macro**, and then click **Visual Basic Editor**.

   This starts the Visual Basic Editor or switches to its window if it's already open.
3. On the **Insert** menu, click **UserForm**.

   A blank form is created and the toolbox is displayed.
4. Right-click the toolbox, and then click **Additional Controls**.

   The **Additional Controls** dialog box is displayed.
5. In the **Available Controls** box, select **Microsoft WebBrowser Control**, and then click **OK**.

   A tool icon is added to the toolbox for the WebBrowser Control. You don't need to repeat steps 4 and 5 the next time you use the toolbox.
6. Click the new tool, and then click the form where you want to place the control.

For more information about UserForms, see Chapter 12, "ActiveX Controls and Dialog Boxes."

{ewc mvimg, mvimage,!tip.bmp}

To display a Web page or document in the WebBrowser control, use the **Navigate** method in Visual Basic. The syntax for the **Navigate** method is:

object.**Navigate** *URL*

Object is either the name of the WebBrowser control on your form or an object variable that refers to it, and *URL* is a string expression that evaluates to a valid URL or path. *URL* can refer to a Web page or other content on the Internet or an intranet, as well as to an Office document, such as a Word document.

If *URL* refers to an Internet protocol and a location on the Internet, the WebBrowser control must establish a connection before is can display the document. If the computer running your application is connected to a *proxy server* (a secure connection to the Internet through a LAN), or if it has a direct connection to the Internet, the WebBrowser control downloads and displays the Web page or other Internet content immediately. If the computer running your application uses a modem and dial-up connection to the Internet, and that connection hasn't been established beforehand, the WebBrowser control initiates the connection. For example, if the user's computer uses a modem and The Microsoft Network to connect to the Internet, the **Sign In** dialog box is displayed to establish the connection to the Internet before the WebBrowser control can display Internet content.

If *URL* refers to an Internet protocol and a location on an intranet server, the computer running your application must be connected to the intranet and have permission to access that server.

If *URL* refers to a standard file system path on a local hard disk or LAN, the WebBrowser control opens the document and displays it immediately. The WebBrowser control can open Office documents (except Microsoft Access databases), text files, and HTML documents that don't require features supported only by an Internet/intranet server. For example, the WebBrowser control can't open and run IDC/HTX files or ASP files from the standard file system, but it can open HTML documents that contain only the HTML tags supported by Microsoft Internet Explorer version 3.0.

Note If *URL* refers to a path in the standard file system that doesn't refer to a file name (for example, C:\Windows\System\), the WebBrowser control displays the file system itself, much like My Computer.

The examples and code in the following sections are specific to developing an application that uses the WebBrowser control in Microsoft Access; however, in most cases, you can apply the same basic principles and techniques to using the WebBrowser control in applications developed with other Office applications.

This section includes the following topics:

By using the WebBrowser control, you can create a Microsoft Access form that performs most of the functions of Microsoft Internet Explorer version 3.0. For example, the following illustration shows the Custom Browse form (WebBrowseWeb) in the Developer Solutions sample application.

{ewc mvimg, mvimage,!illust.bmp}

When a user types a valid URL in the text box at the top of the form (txtLinks) and presses ENTER, the WebBrowser control (`ActiveXCtl1`) displays the Web page or document. Pressing ENTER triggers the AfterUpdate event of the `txtLinks` text box; the AfterUpdate event contains the following code, which goes to the address specified in the URL that the user entered.

```
Private Sub txtLinks_AfterUpdate()
On Error Resume Next
    ' If the user has entered an address (URL) in this control,
    ' attempt to go to the address.
    If Len(Me!txtLinks) > 0 Then
        Me!ActiveXCtl1.Navigate Me!txtLinks
    End If
End Sub
```

Error handling is passed to the control itself because it displays the same error messages displayed by Microsoft Internet Explorer version 3.0.

If you prefer to start navigation by clicking a command button instead pressing ENTER, you can use similar code in the button's Click event.

The Home, Back, Forward, Refresh, and Search buttons on the Custom Browse form use the corresponding **GoHome**, **GoBack**, **GoForward**, **Refresh**, and **GoSearch** methods of the WebBrowser control. For information about how to view brief descriptions about the properties, methods, and events of the WebBrowser control, see Viewing Descriptions of the Properties, Methods, and Events of the WebBrowser Control later in this chapter.

With the Save Location button on the Custom Browse form, you can save the address and a description of the current document to the Links table in the Developer Solutions sample application. When you click the Save Location button, Microsoft Access checks to see if the URL has been saved previously, and if not, uses the following statement to open the **Save Location To Table** dialog box.

```
DoCmd.OpenForm "frmSaveURLDialog", acWindowNormal, , , acFormEdit, acDialog, _
    ctlHyper.LocationName & ";" & ctlHyper.LocationURL
```

The last argument of this statement (ctlHyper.LocationName & ";" & ctlHyper.LocationURL) sets the **OpenArgs** property to a concatenated string that contains the two values returned by the **LocationName** and **LocationURL** properties of the document currently displayed in the Custom Browse form. When the **Save Location To Table** dialog box opens, code in its **Load** event parses the **OpenArgs** property value back into two parts and displays them as the default description and address. When the user clicks **OK**, the description and address in the **Save Location To Table** dialog box form are saved in the Hyperlink and Description fields in the Links table.

For more information about the Custom Browse form, open the Developer Solutions sample application located in the Samples subfolder of your Office folder. To view the Developer Solutions sample application, you must click **Custom** when you install Microsoft Access and then choose to install all sample databases.

By using the WebBrowser control, you can create a Microsoft Access form that displays documents specified in hyperlinks stored in a table. For example, the following illustration shows the Browse Saved Hyperlinks form (WebBrowseTable) in the Developer Solutions sample application. You can use the Browse Saved Hyperlinks form to browse addresses saved in the Links table.

{ewc mvimg, mvimage,!illust.bmp}

When a user clicks a record navigation button at the bottom of the form to move to a new record, the following code in the form's Current event displays the Web page or document whose address is stored in the current record.

```
Private Sub Form_Current()
    Dim varFull As Variant, varDescription As Variant
    Dim HyperlinkAddress As String, HyperlinkSubAddress As String
    Dim msg1 As String, msg2 As String, rst As Recordset, strDisplay As String

    On Error Resume Next

    Set rst = Me.RecordsetClone
    rst.Bookmark = Me.Bookmark
    varFull = rst!HyperLink

    If IsNull(varFull) Then GoTo Current_Err
    varDescription = rst!Description
    Me!ActiveXCtl1.Navigate HyperlinkPart(varFull, acAddress)

    If Err = 438 Then Exit Sub

    gvarBookMark = Me.Bookmark

Current_Bye:
    Exit Sub
Current_Err:

msg1 = "Invalid hyperlink address. Remove the record described as '"
msg2 = "' from the Links table or edit the hyperlink to supply a valid address."

MsgBox msg1 & rst!Description & msg2

        Me.Bookmark = gvarBookMark
        Exit Sub
End Sub
```

This procedure uses the **Navigate** method of the WebBrowser control to display the next hyperlink address. However, don't pass the contents of a Hyperlink field directly to the **Navigate** method. If a user enters or edits data stored in a Hyperlink field from a datasheet or form, it may contain up to three parts of information separated by the pound sign (#). Even if the user doesn't enter all three parts in the datasheet or form, Microsoft Access automatically stores pound signs in the field. If there are pound signs in the Hyperlink field, passing the data from the field directly to the **Navigate** method generates an error. To handle this, the stored value is passed to the **HyperlinkPart** function to extract just the *address* portion of the saved hyperlink, which is then passed to the **Navigate** method. If navigation is successful, the form's **Bookmark** property value is stored in a public variable. This public variable is used to return to the last record if subsequent navigation fails.

Using code to save data in a Hyperlink field doesn't automatically save pound signs in the field. To preserve the proper functioning of a Hyperlink field in other contexts, you may want to write your code to save pound signs before and after a hyperlink address. To see an example of how to do this, view the event procedure set for the Click event of the Save Location button (cmdSaveLocation) on the Custom Browse form.

Note you don't have to store addresses in a Hyperlink field if you don't need users to be able to navigate to

addresses by clicking them in datasheets or forms, or if you don't need to save addresses as HTML anchor tags when saving as HTML. As long as an address doesn't exceed 255 characters, you can store it in a Text field. If an address exceeds 255 characters, you can store it in a Memo field. In either case, you can pass the value stored in the field directly to the **Navigate** method.

For more information about the Browse Saved Hyperlinks form, open the Developer Solutions sample application located in the Samples subfolder of your Office folder. For more information about the format of data stored in a Hyperlink field, see The Hyperlink Field Storage Format earlier in this chapter.

The entire title of this section is *Viewing Descriptions of the Properties, Methods, and Events of the WebBrowser Control*.

Like built-in Office objects, the WebBrowser control has properties that your application can set or read to determine the control's characteristics, methods that your application can use to perform operations on the control, and events your application can respond to. You can view brief descriptions of the properties, methods, and events of the WebBrowser control by using the Object Browser.

Important In order for these properties, methods, and events to appear in the Object Browser, a reference must be set to the **Microsoft Internet Controls** object library. To set this reference, open a module (Microsoft Access) or open the Visual Basic Editor (Microsoft Excel, Word, or PowerPoint), click **References** on the **Tools** menu, and select the **Microsoft Internet Controls** check box in the **Available References** box.

---

u **To view descriptions of the WebBrowser control's methods, properties, and events**

® In Microsoft Excel, Word, or PowerPoint, open the Visual Basic Editor. In Microsoft Access, open a module.

® On the **View** menu, click **Object Browser**.

® In the **Project/Library** box, click **SHDocVw**.

® In the **Classes** box, click **WebBrowser**.

The Members Of box lists the methods, properties, and events associated with the WebBrowser control.

Click here for more information about the methods, properties, and events of the WebBrowser control.
{ewc mvimg, mvimage,!intjump.bmp}

If you purchased Microsoft Office 97 on CD-ROM, you can open a Help file named Iexplore.hlp that contains this information in the \ValuPack\Access\WebHelp folder on the CD-ROM.

Unlike most other ActiveX controls, you can't install the WebBrowser control by itself. For an application that uses the WebBrowser control to work, Microsoft Internet Explorer version 3.0 must also be installed on the computer. Microsoft Internet Explorer version 3.0 can be distributed freely, and doesn't require the payment of royalties or other licensing fees. For information about installing Microsoft Internet Explorer version 3.0, see Adding the WebBrowser Control to a Form earlier in this chapter.

Microsoft Office 97, Developer Edition provides the Internet Transfer control (Msinet.ocx), which you can use to connect to and retrieve files from any Web site that uses either Hypertext Transfer Protocol (HTTP) or File Transfer Protocol (FTP). For example, you could use the Internet Transfer control to:

® Add an FTP browser to any application.

® Create an application that automatically downloads files from a public FTP site.

® Search a World Wide Web site for references to graphics and download only the graphics.

® Retrieve specific pieces of information from a Web page.

Because HTTP and FTP work differently, the operations you can perform with the Internet Transfer control depend on which protocol you are using. For example, the **GetHeader** method only works with HTTP (HTML documents). However, there are a few operations that you can perform with either protocol:

® Set the **AccessType** property of the Internet Transfer control to a valid proxy server.

® Use the **OpenURL** method with a valid URL.

® Use the **Execute** method with a valid URL and command appropriate to the protocol and then use the **GetChunk** method to retrieve data from the buffer.

{ewc mvimg, mvimage,!tip.bmp}

This section includes the following topics:

In Microsoft Excel, Word, and PowerPoint, you can add the Internet Transfer control to a UserForm you create with the Visual Basic Editor. Although the Internet Transfer control is available in the toolbox in Microsoft Excel, Word, and PowerPoint, you can't add the control directly to their documents. In Microsoft Access, you can add the Internet Transfer control to a form in Design view. The Internet Transfer control doesn't display when your application is running.

To add the Internet Transfer control to a Microsoft Excel, Word, or PowerPoint UserForm created with the Visual Basic Editor

1. Open a Microsoft Excel, Word, or PowerPoint document.
2. On the **Tools** menu, point to **Macro**, and then click **Visual Basic Editor**.

   This starts the Visual Basic Editor or switches to its window if it's already open.
3. On the **Insert** menu, click **UserForm**.

   A blank form is created and the toolbox is displayed.
4. Right-click the toolbox, and then click **Additional Controls**.

   The **Additional Controls** dialog box is displayed.
5. In the **Available Controls** box, select **MSInet Control, version 5.0**, and then click **OK**.

   A tool icon is added to the toolbox for the Internet Transfer control. You don't need to repeat steps 4 and 5 the next time you use the toolbox.
6. Click the new tool, and then click the form where you want to place the control.

   By default, the new control is named Inetn, where *n* is some number.

To add the Internet Transfer control to a Microsoft Access form

1. Open the form in Design view.
2. In the toolbox, click the **More Controls** tool.

   A menu appears that lists all the registered ActiveX controls in your system.
3. On the menu of ActiveX controls, click **MSInet Control**.
4. On the form, click where you want to place the control.

   By default, the new control is named ActiveXctln, where *n* is some number.

In order to make any kind of connection to the Internet, you must determine how your computer is connected to the Internet. If you are on an intranet you will probably be connected to the Internet through a *proxy server*.

When using a proxy server, all computers on an intranet that need to connect to the Internet must do so through the proxy server. By using a proxy server, sometimes called a *firewall*, you can protect your local area network from being accessed by others on the Internet. The proxy server acts as a one-way barrier between your internal network and the Internet, preventing others on the Internet from accessing confidential information on your internal network.

u **To determine the proxy server settings on your computer**

Note The following steps apply only to computers running Windows 95 and Windows NT Workstation version 4.0.

---

1. On the **Taskbar** of your computer, click **Start**, point to **Settings**, and then click **Control Panel**.
2. Double-click the I**nternet** icon.
3. In the **Internet Properties** dialog box, click the **Connection** tab.
4. If the **Connect through a proxy server** check box is selected, click **Settings**.
5. The **Proxy Settings** dialog box shows the name of your intranet's proxy server. If no proxy server is defined, contact your workgroup administrator for available proxy servers.

If you want to use a proxy server other than that named in the **Proxy Settings** dialog box, set the **AccessType** property of the Internet Transfer control to **icNamedProxy** (2). Then set the **Proxy** property to the name of the proxy server you want to use.

If you prefer to use the default proxy server, set the **AccessType** property to i**cUseDefault** (0). You don't need to set the **Proxy** property when you use the default proxy server.

The following table describes the settings for the **AccessType** property.

Constant	Value	Description
icUseDefault	0	(Default) The control uses default proxy server settings found in the Windows registry.
IcDirect	1	The control has a direct connection to the Internet.
IcNamedProxy	2	The control uses the proxy server specified in the **Proxy** property.

After you have set the **AccessType** property, the most basic operation is to use the **OpenURL** method with a valid URL to retrieve data on the Internet. When you use the **OpenURL** method, the result depends on the target URL. The following example returns the HTML document found on the Microsoft home page at http://www.microsoft.com to a text box named Text1.

```
' A TextBox control named Text1 contains the
' return result of the method. The Internet Transfer
' control is named Inet1.
Text1.Text = Inet1.OpenURL("http://www.microsoft.com/")
```

In Microsoft Access, a value assigned to the **Text** property can't be longer than 1,024 characters. Substitute the following line of code that sets the **Value** property of the text box instead.

```
Text1.Value = ActiveXCtl0.OpenURL("http://www.microsoft.com/")
```

As a result, the text box displays the HTML source code from the Web site, which may resemble the following illustration.

{ewc mvimg, mvimage,!illust.bmp}

In this case, the default action was to return the HTML document located at the URL. However, if the URL specifies a particular text file, the **OpenURL** method retrieves the actual file. For example, the following code:

```
' In Microsoft Access, substitute Text1.Value
' for Text1.Text in the following line.
Text1.Text = Inet1. OpenURL("ftp://ftp.microsoft.com/disclaimer.txt")
```

retrieves the actual text of the file, as shown in the following illustration.

{ewc mvimg, mvimage,!illust.bmp}

Finally, you can use the **OpenURL** method with a URL that includes extra data appended to it. For example, many Web sites offer the ability to search a database. To search a database from a Web site, you can send a URL that includes the search criteria. The following example uses the search engine at the www.yahoo.com site with the search criteria p=maui.

```
Dim strURL As String

strURL = "http://www.yahoo.com/bin/search.exe?p=maui"
' In Microsoft Access, substitute Text1.Value
' for Text1.Text in the following line.
Text1.Text = Inet1.OpenURL(strURL)
```

If the search engine finds a match for the criteria, the server returns an HTML document that contains the appropriate information.

Saving Text to a File by Using the OpenURL Method

If you want to save retrieved text to a file, use the OpenURL method with the **Open**, **Write**, and **Close** statements, as shown in the following example.

```
Dim strURL As String
Dim intFile As Integer

IntFile = FreeFile()
strURL = "http://www.microsoft.com/"
Open "MSsource.txt" For Output As #IntFile
Write #IntFile, Inet1.OpenURL(strURL)
Close #IntFile
```

You can't save binary files to disk by using the **OpenURL** method. You must use the **Execute** method in

conjunction with the **GetChunk** method as described later in this chapter.

The **OpenURL** method results in a *synchronous* transmission of data. In this context, synchronous means that the transfer operation occurs before any other procedures are run. Thus the data transfer must be completed before you can run any other code.

The **Execute** method, on the other hand, results in an *asynchronous* transmission. When you use the **Execute** method, the transfer operation occurs independently of other procedures. Thus, after the **Execute** method is initiated, other code can run while data is received in the background.

Using the **OpenURL** method results in a direct stream of data that you can save to disk, or view directly in a **TextBox** control (if the data was text). On the other hand, if you are using the **Execute** method to retrieve data, you must monitor the control's connection state by using the StateChanged event. When the appropriate state is reached, use the **GetChunk** method to retrieve data from the control's buffer. This operation is discussed in greater detail in the sections that follow.

You can use the **Execute** method with the FTP and the HTTP protocols to retrieve data or perform operations on Internet servers. The syntax for the **Execute** method is:

controlname.**Execute** *url*, *operation*, *data*, *requestheaders*

The following table describes the arguments of the **Execute** method.

Argument	Description
controlname	Required. The name of the Internet Transfer control you are working with.
url	Optional. Specifies the URL that you want to connect to.
operation	Optional. Specifies the type of operation to perform.
data	Optional. Specifies additional information needed for HTTP GET, HEAD, POST, and PUT methods.
requestheaders	Optional. Specifies additional headers to be sent from the remote server.

This section includes the following topics:

® Using the Execute Method with the FTP Protocol

® Using the Execute Method with the HTTP Protocol

® Using the GetChunk Method

When using the **Execute** method with the FTP protocol, you only use the operation argument, and optionally, the url argument. The url argument is optional because after the first time you invoke the **Execute** method with the url argument, the FTP connection remains open. You can perform additional **Execute** method operations on the same URL until a new URL is specified, or until you perform the CLOSE operation. The following example retrieves a file from a remote computer.

```
Inet1.Execute "FTP://ftp.microsoft.com", _
    "GET disclaimer.txt c:\temp\disclaimer.txt"
```

For FTP operations, you do not use the data and requestheaders arguments. You pass all of the operations and their parameters as a single string in the operation argument, with parameters separated by a space, as follows:

operationname parameter1 parameter2

For example, to retrieve a file, the following code includes the operation name (GET), and the two file names required by the operation.

```
' Get the file named Disclaimer.txt and copy it to the
' location C:\Temp\Disclaimer.txt.
Inet1.Execute, "GET Disclaimer.txt C:\Temp\Disclaimer.txt"
```

The *operationname* part of the *operation* argument is an *FTP command*. If you have used FTP to retrieve files from anonymous FTP servers, you are familiar with commands used to navigate through server trees, and to retrieve files to a local hard disk. For example, to change to a different directory with the FTP protocol, you use the "CD" command with the path to the directory you want to change to.

For the most common operations, such as putting a file on a server and retrieving a file from a server, the Internet Transfer control uses the same or a similar command with the **Execute** method. The following example uses the "CD" command as an argument of the **Execute** method to change to a different directory.

```
' The txtURL text box contains the path to open. The txtRemotePath
' text box contains the path to change to.
Inet1.Execute txtURL.Text, "CD " & txtRemotePath.Text
```

The following table lists the FTP commands that you can use in the operation argument of the **Execute** method.

FTP command	**Description**	**Example**
CD *path*	Change Directory. Changes to the directory specified in *path*.	`Inet1.Execute , "CD docs\mydocs"`
CDUP	Changes to parent directory. Same as “CD ..”	`Inet1.Execute , "CDUP"`
CLOSE	Closes the current FTP connection.	`Inet.Execute , "CLOSE"`
DELETE *file*	Deletes the file specified in *file*.	`Inet1.Execute , _ "DELETE discard.txt"`
DIR *path*	Searches the directory specified in *path*. If *path* isn’t supplied, the current working directory is searched. Use the **GetChunk** method to return the directory listing.	`Inet1.Execute , "DIR /mydocs"`
GET *file1 file2*	Retrieves the remote file specified in *file1*, and creates a new local file specified in *file2*.	`Inet1.Execute , _ "GET getme.txt C:\ gotme.txt"`
MKDIR *path*	Creates a directory as specified in path. Success is dependent on user privileges on the remote host.	`Inet1.Execute , "MKDIR /myDir"`

PUT *file1 file2*	Copies a local file specified in *file1* to the remote host specified in *file2*.	`Inet1.Execute , _ "PUT C:\ putme.txt /putme.txt"`
PWD	Print Working Directory. Returns the current directory name. Use the GetChunk method to return the directory name.	`Inet1.Execute , "PWD"`
QUIT	Terminate current connection	`Inet1.Execute , "QUIT"`
RECV *file1 file2*	Same as GET.	`Inet1.Execute , _ "RECV getme.txt C:\gotme.txt"`
RENAME *file1 file2*	Renames a file. Success is dependent on user privileges on the remote host.	`Inet1.Execute , _ "RENAME old.txt new.txt`
RMDIR *path*	Removes a directory. Success is dependent on user privileges on the remote host.	`Inet1.Execute , "RMDIR oldDir"`
SEND *file*	Copies a file to the remote host. (same as PUT.)	`Inet1.Execute , _ "SEND C:\putme.txt /putme.txt"`
SIZE *file*	Returns the size of the file specified in *file*.	`Inet1.Execute _ "SIZE /largefile.txt"`

Important If your proxy server is a CERN proxy server, you cannot make direct FTP connections by using the **Execute** method. In that case, to get a file, use the **OpenURL** method with the **Open**, **Put**, and **Close** statements, as described in “Saving Text to a File by Using the OpenURL Method” in Using the OpenURL Method, earlier in this chapter. You can also use the **OpenURL** method to get a directory listing by invoking the method and specifying the target directory as the URL.

Logging On to FTP Servers

FTP servers can be either public or private. Anyone can log on to a public server. To log on to a private server, on the other hand, you must be a registered user of the server. In either case, the FTP protocol requires that you supply a user name and a password.

When logging on to public servers, it is common practice to log on as “anonymous,” (UserName = “anonymous”) and use your e-mail name as the password. With the Internet Transfer control, the process of logging on is simplified even further. By default, if you do not specify values for the **UserName** and **Password** properties, the Internet Transfer control uses “anonymous” as your user name, and your e-mail name as the password.

If you are logging on to a private server, set the **UserName**, **Password**, and **URL** properties to appropriate values, and use the **Execute** method, as shown in the following example.

```
With Inet1
    .URL = "ftp://ftp.someFTPSite.com"
    .UserName = "John Smith"
    .Password = "mAuI&9$6"
    .Execute ,"DIR"             ' Returns the directory.
    .Execute ,"CLOSE"           ' Close the connection.
End With
```

After you invoke the **Execute** method, the FTP connection remains open. You can then continue to use the **Execute** method to perform other FTP operations such as CD and GET. When you have completed the session, close the connection by using the **Execute** method with the CLOSE operation. You can also close the

connection automatically by changing the **URL** property, and invoking either the **OpenURL** or **Execute** method; this closes the current FTP connection and opens the new URL.

When you use the **Execute** method with the HTTP protocol to request data from the server, you use the GET, HEAD, POST, and PUT methods in the operation argument. You can use these methods with the **Execute** method, as shown in the following table.

HTTP method	Description	Example
GET	Retrieves the file specified in the *url* argument.	`Inet1.Execute _ "http://www.microsoft.com" & _ "/default.htm", "GET"`
HEAD	Retrieves only the headers of the file specified in the *url* argument.	`Inet1.Execute , "HEAD"`
POST	Provides additional data to support a request to the remote host.	`Inet1.Execute , "POST", strFormData`
PUT	Replaces data at the specified URL.	`Inet1.Execute , "PUT", "replace.htm"`

Using the Execute Method with the Common Gateway Interface

On many World Wide Web sites, you can search a database for criteria that you specify. Most Web sites accomplish this by using the HTTP protocol, which can send queries that use the Common Gateway Interface (CGI).

It is not in the scope of this section to explain the CGI; however, if you are familiar with the CGI, you can use the **Execute** method to construct an application that simulates the search behavior of these Web sites. The following example shows a typical CGI query string.

```
http://www.yippee.com/cgi-bin/find.exe?find=Hangzhou
```

You could send this same query by using the **Execute** method, as follows.

```
Dim strURL As String, strFormData As String

strURL = "//www.yippee.com/cgi-bin/find.exe"
strFormData = "find=Hangzhou"
Inet1.Execute strURL, "POST", strFormData
```

To retrieve resulting data from a server, you must use the **GetChunk** method, as described in the following section.

When you download data from a remote computer by using the **Execute** method, an asynchronous connection is made. For example, if you use the **Execute** method with the HTTP GET method, the server retrieves the requested file. When the entire file has been retrieved, the StateChanged event returns **icResponseCompleted** (12). At that point, you can use the **GetChunk** method to retrieve the data from the buffer. This is shown in the following example.

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
    Dim vtData As Variant            ' Data variable.
    Dim intFile As Integer      ' File number variable.

    intFile = FreeFile()        ' Get free file number.
    Select Case State
    .
    . ' Other cases not shown.
    .
    Case icResponseCompleted
        ' Open a file to write to.
        Open "test.txt" For Binary Access _
            Write As #intFile

        ' Get the first chunk. NOTE: specify a byte
        ' array (icByteArray) to retrieve a binary file.
        vtData = Inet1.GetChunk(1024, icString)

        Do While LenB(vtData) > 0
            Put #intFile, , vtData
            ' Get next chunk.
            vtData = Inet1.GetChunk(1024, icString)
        Loop
        Put #intFile, , vtData
        Close #intFile

    End Select
End Sub
```

Microsoft Office 97, Developer Edition also provides the WinSock control, which you can use to connect to a remote computer and exchange data. You use the WinSock control with either the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP). You can use both protocols to create client and server applications. The WinSock control doesn't have a visible interface at run time.

You can use the WinSock control to:

® Create a client application that collects user information before sending it to a central server.

® Create a server application that functions as a central collection point for data from several users.

® Create an application in which uses can exchange messages in real time, or "chat" with each other.

This section includes the following topics:

To use the WinSock control, you must first decide which protocol to use. The major difference between TCP and UDP is their connection state:

- The TCP protocol requires a persistent connection. It is analogous to a telephone — the user must establish a connection before proceeding.
- The UDP protocol is a connectionless protocol. The transaction between two computers is like passing a note — a message is sent from one computer to another, but there is no persistent connection between the two.

Here are a few questions that may help you determine which protocol to use:

- Will the application require acknowledgment from the server or client when data is sent or received? If so, use the TCP protocol because it requires an explicit connection before sending or receiving data.
- Is the integrity of your data critical? If so, use the TCP protocol. Once a connection has been made, the TCP protocol maintains the connection and ensures the integrity of the data. If the integrity of your data is not critical, you can improve performance by using the UDP protocol. Using the UDP protocol can be faster and uses less network bandwidth, but you may experience a certain amount of data loss. However, when transmitting an image or a sound file, the data loss may not even be noticeable.
- Will the data be sent intermittently or in one session? If the data will be sent intermittently, you may want to use the UDP protocol because it requires fewer network resources. For example, use the UDP protocol if you are creating an application that notifies specific computers when certain tasks have completed. If you want the data to be sent in one session, use the TCP protocol because it maintains a persistent connection to the network.

In Microsoft Excel, Word, and PowerPoint, you can add the WinSock control to a UserForm you create with the Visual Basic Editor. Although the WinSock control is available in the **Control Toolbox** in Microsoft Excel, Word, and PowerPoint, you can't add the control directly to their documents. In Microsoft Access, you can add the WinSock control to a form in Design view. The WinSock control doesn't display when your application is running.

u **To add the WinSock control to a Microsoft Excel, Word, or PowerPoint UserForm created with the Visual Basic Editor**

1. Open a Microsoft Excel, Word, or PowerPoint document.
2. On the **Tools** menu, point to **Macro**, and then click **Visual Basic Editor**.

   This starts the Visual Basic Editor or switches to its window if it's already open.
3. On the **Insert** menu, click **UserForm**.

   A blank form is created and the toolbox is displayed.
4. Right-click the toolbox, and then click **Additional Controls**.

   The **Additional Controls** dialog box is displayed.
5. In the **Available Controls** box, select **WinSock Control, version 5.0**, and then click **OK**.

   A tool icon is added to the toolbox for the WinSock control. You don't need to repeat steps 4 and 5 the next time you use the toolbox.
6. Click the new tool, and then click the form where you want to place the control.

   By default, the new control is named Winsockn, where n is some number.

u **To add the WinSock control to a Microsoft Access form**

1. Open the form in Design view.
2. In the toolbox, click the **More Controls** tool.

   A menu appears that lists all the registered ActiveX controls in your system.
3. On the menu of ActiveX controls, click **WinSock Control, version 5.0**.
4. On the form, click where you want to place the control.

   By default, the new control is named ActiveXctl*n*, where *n* is some number.

After you add the WinSock control to your form, you specify which protocol you are going to use. If you want to use the UDP protocol, set the **Protocol** property to **sckUDPProtocol**. The default setting of the **Protocol** property is **sckTCPProtocol**. You can set the **Protocol** property in the property sheet or in Visual Basic code, as follows.

```
Winsock1.Protocol = sckUDPProtocol
```

To connect to a remote computer, you must know either its Internet Protocol (IP) address or its “friendly name.” The IP address is a series of three digit numbers separated by periods (*nnn.nnn.nnn.nnn*). It’s much easier to remember the friendly name of a computer.

To determine the name of a computer

1. On the Taskbar of your computer, click Start, point to Settings, and then click Control Panel.
2. Double-click the Network icon.
3. Click the Identification tab.
4. The name of your computer is in the Computer name box.

After you have determined a computer’s name, you can use it as the value for the **RemoteHost** property of a WinSock control, as shown in the examples later in this section.

When creating an application that uses the TCP protocol, you must first decide if your application will be a client or a server. The client makes a connection request, which the server can then accept to complete the connection. After the connection is complete, the client and server can freely communicate with each other.

To create a TCP server

1. Create a Microsoft Excel, Word, or PowerPoint document or a Microsoft Access database.
2. Create a form and name it frmServer.
3. Set the Caption property of the form to TCP Server.
4. Add a WinSock control to the form and set its Name property to tcpServer.
5. Add two text box controls to the form. Name the first txtSendData, and the second txtOutput.
6. Add the following code to the form.

```
Private Sub Form_Load()
    ' Set the LocalPort property to an integer.
    ' Then invoke the Listen method.
    tcpServer.LocalPort = 1001
    tcpServer.Listen
End Sub

Private Sub tcpServer_ConnectionRequest (ByVal requestID As Long)
    ' Check if the value of the control's State property
    ' is closed. If not, close the connection before
    ' accepting the new connection.
    If tcpServer.State <> sckClosed Then tcpServer.Close
    ' Accept the request with the requestID parameter.
    tcpServer.Accept requestID
End Sub

Private Sub txtSendData_Change()
    ' The TextBox control named txtSendData
    ' contains the data to be sent. Whenever the user
    ' types into the textbox, the string is sent
    ' using the SendData method.
    tcpServer.SendData txtSendData.Text
End Sub

Private Sub tcpServer_DataArrival (ByVal bytesTotal As Long)
    ' Declare a variable for the incoming data.
    ' Use the GetData method and set the Text
    ' property of a TextBox named txtOutput to
    ' the data.
    Dim strData As String
    tcpServer.GetData strData
    ' In Microsoft Access, substitute txtOutput.Value
    ' for txtOutput.Text in the following line.
    txtOutput.Text = strData
End Sub
```

These procedures create a simple server application. To complete the scenario, you must also create a client application.

To create a TCP client

1. Create a form and name it frmClient.
2. Set the **Caption** property of the form to TCP Client.
3. Add a WinSock control to the form and set its **Name** property to tcpClient.

4. Add two text box controls to the form. Name the first txtSendData, and the second txtOutput.
5. Add a command button control to the form and name it cmdConnect.
6. Set the **Caption** property of the command button control to Connect.
7. Add the following code to the form.

Important Set the value of the **RemoteHost** property to the name of your computer.

```
Private Sub Form_Load()
    ' The name of the Winsock control is tcpClient.
    ' Note: To specify a remote host, you can use
    ' either the IP address (ex: "121.111.1.1") or
    ' the computer's friendly name, as shown here.
    tcpClient.RemoteHost = "RemoteComputerName"
    tcpClient.RemotePort = 1001
End Sub

Private Sub cmdConnect_Click()
    ' Invoke the Connect method to initiate a
    ' connection.
    tcpClient.Connect
End Sub

Private Sub txtSendData_Change()
    tcpClient.SendData txtSendData.Text
End Sub

Private Sub tcpClient_DataArrival _
(ByVal bytesTotal As Long)
    Dim strData As String
    tcpClient.GetData strData
    ' In Microsoft Access, substitute txtOutput.Value
    ' for txtOutput.Text in the following line.
    txtOutput.Text = strData
End Sub
```

The preceding code creates a simple client/server application. To try the two together, make a copy of the application and put it on another computer. Open the client on one computer and open the server on the other computer. Then click **Connect** on the client form. When you type text into the txtSendData text box on either form, the same text appears in the txtOutput text box on the other form.

Accepting More Than One Connection Request

With Microsoft Word, Microsoft Excel, Microsoft PowerPoint, and Microsoft Access forms, you can only create a server that accepts only one connection request. However, you can use Microsoft Visual Basic version 4.0 or later to create a server application that accepts several connection requests by using the same control. To do so, you create a new instance of the control by setting its **Index** property; this creates a control array. Then you invoke the **Accept** method on the new instance. You do not need to close the connection.

The following code assumes there is a WinSock control on a form named sckServer, and that its **Index** property has been set to 0; thus the control is part of a control array. In the Declarations section, a module-level variable intMax is declared. In the form's Load event, intMax is set to 0, and the **LocalPort** property for the first control in the array is set to 1001. Then the **Listen** method is invoked for the control, making it the control that receives connection requests. As each connection request arrives, the code tests it to see if the **Index** property is 0 (the value of the "listening" control). If so, the listening control increments intMax, and uses that number to create a new control instance. The new control instance then accepts the connection request.

```
Private intMax As Long
```

```
Private Sub Form_Load()
    intMax = 0
    sckServer(0).LocalPort = 1001
    sckServer(0).Listen
End Sub

Private Sub sckServer_ConnectionRequest _
    (Index As Integer, ByVal requestID As Long)
    If Index = 0 Then
        intMax = intMax + 1
        Load sckServer(intMax)
        sckServer(intMax).LocalPort = 0
        sckServer(intMax).Accept requestID
        Load txtData(intMax)
    End If
End Sub
```

Creating a UDP application is even simpler than creating a TCP application because the UDP protocol doesn't require a connection. After you create the forms, add the WinSock controls, and set the **Protocol** property to UDPProtocol, you add code on both computers that performs the following steps:

1. Set the **RemoteHost** property of the WinSock control to the name of the other computer.
2. Set the **RemotePort** property of the WinSock control to the **LocalPort** property of the other WinSock control.
3. Use the **Bind** method to specify the local port to be used by the WinSock control.

The **Bind** method reserves a local port for use by the WinSock control. For example, when you bind the control to port number 1001, no other application can use that port to receive connection requests. This may be useful if you want to prevent another application from using that port.

If there is more than one network adapter on the machine, you can specify which adapter to use in the *LocalIP* argument the **Bind** method. If you do not specify which network adapter to use, the control uses the first adapter listed in the **Network** dialog box, which is available through the computer's Control Panel.

When using the UDP protocol, you can change the setting of the **RemoteHost** and **RemotePort** properties while remaining bound to the same local port. However, with the TCP protocol, you must close the connection before changing the **RemoteHost** and **RemotePort** properties.

In the TCP application created in the previous section, you must set the WinSock control on the client to receive connection requests, and the WinSock control on the server must initiate a connection. In contrast, the two computers in a UDP application do not have such restrictive roles. Both computers can send and receive messages. Because both computers can be considered equal in the relationship, a UDP application is sometimes called a *peer-to-peer application*.

The following procedures create a UDP application that two people can use to exchange messages in real time, or "talk" to each other.

To create a UDP Peer

1. Create a document in Microsoft Excel, Word, or PowerPoint, or create a database in Microsoft Access.
2. Create a form and name it frmPeerA.
3. Set the **Caption** property of the form to Peer A.
4. Add a WinSock control to the form and set its **Name** property to udpPeerA.
5. Set the **Protocol** property to UDPProtocol.
6. Add two text box controls to the form. Name the first txtSendData, and the second txtOutput.
7. Add the following code to the form.

```
Private Sub Form_Load()
    ' The control's name is udpPeerA.
    With udpPeerA
        .Protocol = sckUDPProtocol   ' Set the control to UDP protocol.
        .RemoteHost= "PeerB"         ' Set RemoteHost property to the
                                            ' name of the other computer.
        .RemotePort = 1001              ' Port to connect to.
        .Bind 1002                      ' Bind to the local port.
    End With
    frmPeerB.Show                       ' Show second form.
End Sub

Private Sub txtSendData_Change()
    ' Send text as soon as it's typed.
    udpPeerA.SendData txtSendData.Text
End Sub

Private Sub udpPeerA_DataArrival (ByVal bytesTotal As Long)
    Dim strData As String
```

```
    udpPeerA.GetData strData
    ' In Microsoft Access, substitute txtOutput.Value
    ' for txtOutput.Text in the following line.
    txtOutput.Text = strData
End Sub
```

To create a second UDP Peer

1. Create a form and name it frmPeerB.
2. Set the **Caption** property of the form to Peer B.
3. Add a WinSock control to the form and set its **Name** property to udpPeerB.
4. Set the **Protocol** property to UDPProtocol.
5. Add two text box controls to the form. Name the first txtSendData, and the second txtOutput.
6. Add the following code to the form.

```
Private Sub Form_Load()
    ' The control's name is udpPeerB.
    With udpPeerB
        .Protocol = sckUDPProtocol   ' Set the control to UDP protocol.
        .RemoteHost= "PeerA"         ' Set RemoteHost property to the
                                            ' name of the other computer.
        .RemotePort = 1002              ' Port to connect to.
        .Bind 1001                      ' Bind to the local port.
    End With
End Sub

Private Sub txtSendData_Change()
    ' Send text as soon as it's typed.
    udpPeerB.SendData txtSendData.Text
End Sub

Private Sub udpPeerB_DataArrival (ByVal bytesTotal As Long)
    Dim strData As String

    udpPeerB.GetData strData
    ' In Microsoft Access, substitute txtOutput.Value
    ' for txtOutput.Text in the following line.
    txtOutput.Text = strData
End Sub
```

To try this example, make a copy of the application and put it on another computer. Open the first peer on one computer and open the second peer on the other computer. When you type text into the txtSendData text box on either form, the same text appears in the txtOutput text box on the other form.

Microsoft provides two products that make it easy to create a personal Web server on your computer for low-volume Web publishing: Microsoft Personal Web Server and Microsoft Peer Web Services. These products are ideal for publishing departmental home pages, personal home pages, or small-scale Web applications on your company's intranet.

Although Personal Web Server and Peer Web Services are intended for small-scale Web publishing, they provide most of the same services and features as Microsoft Internet Information Server, a robust Web server intended for high-volume Web publishing. You can use Personal Web Server or Peer Web Services to develop and test Web applications, and then transfer them to a Web server running Microsoft Internet Information Server.

Both Personal Web Server and Peer Web Services can:

® Publish Web pages on the Internet or over a LAN on an intranet by using the HTTP service.

® Support Microsoft ActiveX controls.

® Transmit or receive files by using the FTP service.

® Run Internet Server API (ISAPI) and Common Gateway Interface (CGI) scripts.

® Send queries to ODBC data sources by using the Internet Database Connector component (Httpodbc.dll).

® Support the Secure Sockets Layer.

In addition, Peer Web Services can:

® Use pass-through security to Windows NT Server and Novell NetWare as long as File and Printer Sharing services are installed.

® Use local-user security if Microsoft File and Print Sharing services are not installed.

® Perform remote administration by using a Web-based application.

Note Before you can use Active Server Pages with Microsoft Peer Web Services or Microsoft Personal Web Server, you must download and install the Active Server Pages components after installing your personal web server software. To do this, connect to http://www.microsoft.com/iis/, choose to download IIS 3.0, register to download, and then select to download the Active Server Pages components. After installing the Active Server Pages components, you can read HTML online documentation for the Active Server Pages components at http://MyServant/IASDocs/roadmap.asp.

This section includes the following topics:

To run Personal Web Server or Peer Web Services, you must meet the following installation requirements.

Personal Web Server

® A computer with Windows 95 installed.

® A CD-ROM drive for the installation compact disc.

® Adequate disk space for your information content.

Peer Web Services

® A computer with Windows NT Workstation version 4.0 installed.

® A CD-ROM drive for the installation compact disc.

® Adequate disk space for your information content. It is recommended that all drives used with Peer Web Services be formatted with the Windows NT File System (NTFS).

When using Personal Web Server or Peer Web Services, each computer you want to access the server must have Transmission Control Protocol/Internet Protocol (TCP/IP) installed. The TCP/IP protocol is included with Windows 95 and Windows NT Workstation version 4.0. To install and configure the TCP/IP protocol and related components, double-click the **Network** icon in Control Panel. Each system must meet additional requirements depending on whether you want to use the server on an intranet or the Internet.

Intranet Publication Requirements

® A network adapter card and local area network (LAN) connection.

® The Windows Internet Name Service (WINS) server or the Domain Name System (DNS) server installed on a computer in your intranet. WINS and DNS run only on Windows NT Server. This step is optional, but it does allow users to use “friendly names” instead of IP addresses when connecting to your server.

Internet Publication Requirements

® An Internet connection and Internet Protocol (IP) address from your Internet Service Provider (ISP).

® DNS registration for that IP address. This step is optional, but it does allow users to use “friendly names” instead of IP addresses when connecting to your server. For example, “microsoft.com” is the friendly domain name registered to Microsoft. Within the microsoft.com domain, Microsoft has named its World Wide Web (WWW) server “www.microsoft.com.” Most ISPs can register your domain names for you.

® A network adapter card suitable for your connection to the Internet.

The Setup file for the Personal Web Server is available on the Web. To download the Setup program for Personal Web Server for Windows 95, connect to the Microsoft Personal Web Server home page at:

http://www.microsoft.com/ie/iesk/pws.htm

You can install Personal Web Server if you are running Windows 95 or Windows NT Workstation version 4.0. However, if you are using Windows NT, it is recommended that you install Peer Web Services instead.

u **To install Personal Web Server**

1. Connect to the Personal Web Server home page on the Web and download PWS10a.exe.
2. Double-click **PWS10a.exe**.
3. This starts the installation process. You may be required to supply additional files from your Windows 95 Setup disks.
4. When installation is finished, the Setup program asks if you want to restart your computer. Click **Yes**.

The files to install Peer Web Services are provided on the Microsoft Windows NT Workstation version 4.0 Setup CD-ROM.

u **To install Peer Web Services**

1. Open **Control Panel**, and then double-click **Network**.
2. Click the **Services** tab, and then click **Add**.
3. In the **Network Service** list, double-click **Peer Web Services**.

   This starts the installation process. You may be required to supply additional files from your Windows NT Setup disks.
4. In the first **Microsoft Peer Web Services Setup** dialog box, click **OK**.
5. In the second **Microsoft Peer Web Services Setup** dialog box, select which services you want to install, and then click **OK**.
6. In the **Publishing Directories** dialog box, specify the directories you want to use for each service, or accept the default directories, and then click **OK**.

For more information about Using Personal Web Server or Peer Web Services, you can refer to their online documentation, which is available once installation is complete.

u **To view documentation for Personal Web Server or Peer Web Services**

1. Start your Web browser.
2. To view the documentation for Personal Web Server, in your browser's address box, type:

   http://*MyServer*/docs/default.htm

   To view the documentation for Peer Web Services, in your browser's address box, type:

   http://*MyServer*/iisadmin/htmldocs/inetdocs.htm

   where MyServer is the name of the computer on which you installed Personal Web Server or Peer Web Services. To determine the name of the computer, open Control Panel, double-click the **Network** icon, and then click the **Identification** tab.
4. Press ENTER.

Click here to connect to the Visual Tools page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

If you're making strategic purchase decisions for software development tools, looking for the best tools and technologies to solve a business problem, or need clear descriptions of technical terms, the Microsoft Visual Tools Web site can help. This site presents Microsoft's overall development strategy and the latest Microsoft Visual Tools for building solutions that integrate the Internet and client/server.

You can view an HTML reference that describes the most commonly used HTML tags as well as recent additions supported by Microsoft Internet Explorer and Netscape Navigator at http://www.microsoft.com/workshop/author/newhtml/.

In Word, you can use the **Add** method to add a hyperlink to either a **Range** object (a range or selection of text), a **Shape** object (a graphic object), or an **InlineShape** object (a graphic object within a line of text).

Creating a Hyperlink for a Microsoft Word Range Object

The following example inserts the text “Microsoft Web Site” at the beginning of the active document, selects the inserted text, and then adds a hyperlink to the text that goes to the Microsoft home page at http://www.microsoft.com/.

```
Sub AddHyperlinkRange()
    Dim r As Range

    Set r = ActiveDocument.Range(Start := 0, End := 0)
    r.InsertBefore "Microsoft Web Site"
    Selection.MoveRight Unit := wdWord, Count := 3, Extend := wdExtend
    ActiveDocument.Hyperlinks.Add Anchor := Selection.Range, _
        Address := "http://www.microsoft.com/"
End Sub
```

Creating a Hyperlink for a Microsoft Word Shape Object

The following example creates a beveled shape, adds the text “Microsoft Web Site” to the shape, and then adds a hyperlink to the shape that goes to the Microsoft home page at http://www.microsoft.com/.

```
Sub AddHyperlinkShape()
    ActiveDocument.Shapes.AddShape(msoShapeBevel, 150, 150, 100, 30).Select
    With Selection
        .ShapeRange.TextFrame.TextRange.Select
        .Collapse
        .TypeText Text:="Microsoft Web Site"
        .ShapeRange.Select
    End With
    ActiveDocument.Hyperlinks.Add Anchor:=Selection.ShapeRange, _
        Address:= "http://www.microsoft.com/"
End Sub
```

Creating a Hyperlink Associated with a Command Button

In addition to creating **Hyperlink** objects in Visual Basic code, you can create a command button by using the user interface and then add code to the command button’s Click event procedure to make it follow a hyperlink. This doesn’t create a **Hyperlink** object so the hyperlink isn’t available in the document’s **Hyperlinks** collection. To create a command button that follows a hyperlink in Microsoft Excel, Word, and PowerPoint, use the following procedure.

To create a command button that follows a hyperlink

1. Right-click the menu bar and then click **Control Toolbox** on the shortcut menu.
2. In the toolbox, click the **Command Button** tool, and then click where you want to place the command button.
3. Right-click the command button, and then click **Properties** on the shortcut menu.
4. In the **Caption** property box, enter the text you want on the button. Set any other properties you want to control the button’s appearance and then close the **Properties** dialog box.
5. Right-click the command button, and then click **View Code**. Enter a procedure that uses the **FollowHyperlink** method in the button’s Click event. For example:

```
Private Sub CommandButton1_Click()
    FollowHyperlink "http://www.microsoft.com/"
End Sub
```

6. Save the code, and exit Design mode to test the button.

Note When using this method, the command button doesn't display blue underlined text or the hand cursor when the mouse is over the button.

For more information about the **FollowHyperlink** method, see The FollowHyperlink Method later in this chapter.

In Microsoft Excel, you can use the Add method to add a hyperlink to either a **Range** object (a range of one or more cells) or a **Shape** object (a graphic).

Creating a Hyperlink for a Microsoft Excel Range Object

The following example adds the display text “MSN Web site” to cell A1 in the first worksheet in the current workbook, and then adds a hyperlink to that range that goes to the Web site at http://www.msn.com/.

```
Sub AddHyperlink_Range()
    Dim wrk As Worksheet

    Set wrk = ActiveWorkbook.Sheets(1)
    wrk.Range("A1").Value = "MSN Web site"
    wrk.Hyperlinks.Add Address := "http://www.msn.com/", _
        Anchor := wrk.Range("A1")
End Sub
```

Creating a Hyperlink for a Microsoft Excel Shape Object

The following example adds a rounded rectangle labeled “Click Here” to the first worksheet in the current workbook, and then adds a hyperlink to the rectangle that goes to cell C6 on the first sheet of Book2.xls.

```
Sub AddHyperlink_Shape()
    Dim wrk As Worksheet
    Dim shp As Shape

    Set wrk = ActiveWorkbook.Sheets(1)
    Set shp = wrk.Shapes.AddShape(msoShapeRoundedRectangle, 50, 50, 100, 50)
    shp.Select
    Selection.Characters.Text = "Click Here"
    wrk.Hyperlinks.Add Anchor := shp, Address := "C:\My Documents\Book2.xls", _
        SubAddress := "Sheet1!C6"
End Sub
```

PowerPoint doesn't use the **Add** method to create a new hyperlink. Instead, you create a hyperlink by working with the **ActionSettings** collection of a **Shape** object (a graphic) or a **TextRange** object (text associated with a **Shape** object). A **Shape** or **TextRange** object can have two different hyperlinks assigned to it: one that's followed when the user clicks the object during a slide show, and another that's followed when the user passes the mouse pointer over the object during a slide show.

To specify which mouse action to work with, first use the ActionSettings property to return the **ActionSettings** collection, then use **ActionSettings**(index), where index is either **ppMouseClick** or **ppMouseOver**. Set the **Action** property to **ppActionHyperlink** to specify that the action is a hyperlink. After a hyperlink is created, it's available from the **Hyperlinks** collections for the **Shape**, **TextRange**, and **Slide** objects.

Creating a Hyperlink for a Microsoft PowerPoint Shape Object

The following example adds a Custom action button with text that reads "Microsoft.com" to the first slide in the active presentation, and then adds a hyperlink to the button that goes to the Microsoft home page.

```
Sub AddHyperlinkButton()
    Dim sld As Slide, shp As Shape

    Set sld = ActivePresentation.Slides(1)
    Set shp = sld.Shapes.AddShape(msoShapeActionButtonCustom, 50, 50, 160, 30)
    With shp.TextFrame
        .TextRange.Text = "Microsoft.com"
        .MarginBottom = 5
        .MarginLeft = 5
        .MarginRight = 5
        .MarginTop = 5
    End With
    With shp.ActionSettings(ppMouseClick)
        .Action = ppActionHyperlink
        .Hyperlink.Address = "http://www.microsoft.com/"
    End With
End Sub
```

Creating a Hyperlink for a Microsoft PowerPoint TextRange Object

The following example adds a rectangle to the first slide in the active presentation, adds text to the rectangle, and then adds a hyperlink to the text. This example defines a hyperlink for all the text in the text range. It is possible to define more than one hyperlink within a text range for selected characters.

```
Sub AddHyperlinkText()
    Dim sld As Slide, shp As Shape, txt As Text

    Set sld = ActivePresentation.Slides(1)
    Set shp = sld.Shapes.AddShape(msoShapeRectangle, 0, 0, 250, 140)
    shp.TextFrame.TextRange.Text = "Microsoft Web Site"
    Set txt = shp.TextFrame.TextRange
    With txt.ActionSettings(ppMouseClick)
        .Action = ppActionHypertext
        .Hyperlink.Address = "http://www.microsoft.com/"
    End With
End Sub
```

Microsoft Access doesn't provide a **Hyperlinks** collection or use the **Add** method to create a hyperlink on a form or report. Instead, you create hyperlinks for label, command button, and image controls by setting either the **HyperlinkAddress** property or the **HyperlinkSubAddress** property, or both, of the control.

Note You can also create a field with the Hyperlink data type to store hyperlink addresses in a table, and then bind that field to a text box, list box, or combo box on a form. For more information, see Storing Hyperlinks in Microsoft Access Tables later in this chapter.

Creating a Hyperlink Control in Microsoft Access

The following example creates a new label on a form and then sets the **HyperlinkAddress** and **HyperlinkSubAddress** properties to create a hyperlink. When you create a hyperlink in Visual Basic, and you want it to be colored and underlined, you must also explicitly set the **ForeColor** and **FontUnderline** properties.

```
Sub CreateHyperlinkLabel(strForm As String, xPos As Integer, _
        yPos As Integer, strCaption As String, Optional strAddress As String, _
        Optional strSubAddress As String)
    Dim ctlLabel As Control

    ' Open form, hidden in Design view.
    DoCmd.OpenForm strForm, acDesign,,,,acHidden

    ' Create label control with text specified by strCaption, at
    ' the position specified by xPos and yPos.
    Set ctlLabel = CreateControl(strForm, acLabel, , "", _
        strCaption, xPos, yPos)

    ' Set hyperlink address, text color, and underline.
    With ctlLabel
        .HyperlinkAddress = strAddress
        .HyperlinkSubAddress = strSubAddress
        .ForeColor = "1279872587"
        .FontUnderline = True
    End With

    ' Save form.
    DoCmd.Save acForm, strForm
End Sub
```

To use this example to create a hyperlink, you must specify the form, position, display text, and hyperlink address. For example, enter the following code into the Debug window:

```
CreateHyperlinkLabel "Form1",100,100,"Microsoft Web
Site","http://www.microsoft.com/"
```

You can use similar code to create image and command button controls and set properties to create a hyperlink.

If you use the **FollowHyperlink** method to add hyperlinks to controls that don't support the **HyperlinkAddress** or **HyperlinkSubAddress** properties, you fail to provide any feedback to the user indicating that the control can follow a hyperlink.

One way to inform a user that the control contains a hyperlink is to set the control's **ControlTipText** property so that a text message appears when the user rests the pointer on the control.

By Shamir Dasgupta
Published by The Cobb Group

One technique you can use to spice up static Web pages is to add some interactivity and let the visitor participate. You can encourage visitor participation by incorporating scripts into the page. So far, we've seen examples of CGI (Common Gateway Interface) scripts written in PERL or TCL, which work in the server side, and client-side scripts, which the browser interprets. Both of these methods have drawbacks. The former method isn't object-oriented by nature, so you're limited in what you can accomplish. The latter method is robust and object-oriented, but the user's browser must support it.

When you create your pages in ASP, you have best of both worlds. You can use powerful scripting languages such as VBScript or JavaScript, but you can run the script on the server. When a script is executed on Active Server, the client or the browser receives only the results — not the script itself. For example, if you write a script that determines output based on the user's input, the browser sees only the result. In addition, the result appears in plain HTML, so any HTML-compliant browser can view it. In this article, we'll explore server-side scripting in ASP.

Lunch Plans

For our example, we want to create a simple application with VBScript. The objective of our script is to collect data from our user and display it back in plain HTML. When the page first loads, we'll display a form on which the user will input his or her name, the person he or she prefers to have lunch with, and the reason why. We'll build error-handling right into the script, so if the user leaves any fields empty, VBScript will display a specific message.

Our data-collection form will look like **Figure A**. After the user completes and submits the form, the resulting page will look like **Figure B**. We'll use Visual InterDev's built-in HTML source editor to create the page.

{ewc mvimg, mvimage,!illust.bmp}
Figure A. The initial page simply collects the user's data.

{ewc mvimg, mvimage,!illust.bmp}
Figure B. The results are displayed in plain HTML.

The Basics

Start Visual InterDev and open a Web site you've already created, or create a new Web site. Select New from the File menu; Visual InterDev will display the New dialog box. Highlight Active Server Page; enter *LunchPlan* in the File Name input box, as shown in **Figure C**; and click OK.

{ewc mvimg, mvimage,!illust.bmp}
Figure C. You can create a new Active Server Page using the New dialog box

You'll notice that Visual InterDev creates the new page and adds it to your existing Web site. The source code for the new file LunchPlan.asp now appears in Visual InterDev's right pane, as **Figure D** shows. This is a blank or shell page containing the minimum HTML code that the browser requires.

{ewc mvimg, mvimage,!illust.bmp}
Figure D. You can edit HTML source code directly within Visual InterDev.

You can see the significant difference between this page and any other HTML page illustrated in the very first line of code:

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

When you use client-side scripts, you must enclose your code between the <SCRIPT> and </SCRIPT> tags. Browsers that support the script can then interpret the code. By contrast, the first statement in an Active Server Page designates the scripting language used in the entire page — in our case, VBScript. VBScript is Visual InterDev's default scripting language; however, you can change the default to JavaScript if you prefer.

Now, when we write the script code, we'll simply enclose it within the <% and %> symbols. (You'll notice that Visual InterDev's source editor highlights the scripts in bright yellow for distinction.) Since we've finished the basic process to create a blank Active Server Page, we're ready to write the script.

It’s All in the Script

To collect the user’s data, we’ll need to first create a form, then write the script to perform the form action. To create the form, use the HTML code shown in **Listing A**.

Listing A. Creating a form

```
<form method=“POST” action=“LunchPlan.asp”>
  <p>Your Name: </p>
  <p><input type=“TEXT” size=“50" maxlength=“50" name=“name”><br>
  </p>
  <p>People you’d like to have lunch with (you may select more than one):<br>
  <select name=“LunchPlan” multiple size=“3">
  <option selected> Bill Gates
    </option>
  <option> Larry Ellison
    </option>
  <option> Gil Amelio
    </option>
  </select>
    <input type=“HIDDEN” name=“hname” value=“hvalue”>
  <br>
  </p>
  <p>Why do you want to have lunch with these people? </p>
  <p><textarea name=“describe” rows=“5" cols=“35"></textarea><br>
  </p>
  <p><input type=“SUBMIT” value=“Submit Form”>
<input
  type=“RESET” value=“Reset Form”> </p>
</form>
```

Next, let’s add the script that displays the form. To do so, add the VBScript code from **Listing B**.

Listing B. Displaying the form

```
<%
On Error Resume Next
If Request.Form(“hname”) = “” Then
  ‘ This part of the script allows the visitor
  ‘ to enter data on an HTML form.
%>
```

If you’re familiar with client-side scripting, you’ll notice that nothing is out of the ordinary about the VBScript code itself. However, the implementation may be slightly different from what you’re accustomed to.

The next step is to display the new form based on the user’s input and show alternative text if the user left a data field empty. Use the code from **Listing C** after the form to do so.

Listing C. Displaying user selections

```
<% Else
  ‘ This part of the script shows the visitor
  ‘ what he/she selected.
%>
<% If Request.Form(“name”) = “” Then %>
<p>You did not provide your name.
<% Else %> </p>
<p>Your name is <b><%= Request.Form(“name”) %></b>
<% End If %>
```

```
<% If Request.Form("LunchPlan").Count = 0 Then %> </p>
<p>You did not select anyone.
<% Else %> </p>
<p>The people you want to have lunch with are: <b><%= Request.Form("LunchPlan")
%></b>
<% If Request.Form("describe") = "" Then %> </p>
<p>You did not say why you like the people you've selected.
<% Else %> </p>
<p>Your description of why you want to have lunch with these people: <b><i>
<%= Request.Form("describe") %></b></i>
<% End If %>
<% End If %>
<% End If %>
```

So far, we've created the form and written the script to add some interactivity. Now it's time to test our new Active Server Page. Save the file by selecting Save from the File menu, then open the page in Internet Explorer.

And Now the Difference

Before we test the functionality of the page, let's take a look at the source code from the browser. Select Source from the View menu within Internet Explorer. You'll notice that the VBScript code isn't visible.

To test our new page, we need to enter some data. Let's assume that our user, Scott, wants to have lunch with Bill Gates because he needs help with Java. After you enter this information, click Submit to see the results, as shown in **Figure B** above. View the source code again by selecting Source from the View menu. The source code should look like that shown in **Listing D**.

Listing D. Results source code

```
<html>

<head>
<meta name="GENERATOR" content="Microsoft Visual InterDev 1.0">
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<title>ASP Demo</title>
</head>
<body bgcolor="#ffffff" text="#000000">
<h1>Lunch request</h1>
<hr>
</p>

<p>Your name is <b>Scott</b>  </p>

<p>The people you'd like to have lunch with are: <b>Bill Gates</b>  </p>

<p>Your description of why you want to have lunch with these people: <b><i>I need
help with Java</b></i>

<br>
<br>
</p>
</body>
</html>
```

As you can see, the dynamically generated page includes only the results in plain HTML. The source code isn't exposed, and any HTML-compliant browser can view the results — even if the browser doesn't support VBScript.

Conclusion

Active Server Pages provides robust server-side scripting that you can use to add interactivity to your site. Since the script is executed on the server side, you won’t have to worry about browser compatibility.

Shamir Dasgupta is editor-in-chief of *Microsfot Web Builder*, The Cobb Group's journal for Web developers. His personal Website, www.xprssions.com, won three categories, including Grand Prize, in Microsoft's inaugural *Activate the Internet* competition in 1996.

By Shamir Dasgupta
Published by The Cobb Group

If you want to use a database as your source of information in a traditional Web development environment, you must write a CGI application that can read and write to the database. Common CGI scripting languages such as PERL can run as external applications on your server and retrieve or update the data. However, this method is slow and offers no visual development tools. Moreover, this technique doesn't create an integrated server object. Fortunately, you can eliminate all of these complications by building your site with Active Server Pages (ASP).

ASP executes the component on the Web server and returns just the results, in plain HTML format, to the user's browser. ASP does this by using a very important Active Server component called Active Data Objects (ADO). ADO can access any ODBC-compliant database, such as Microsoft SQL server, using the ActiveX Scripting model. ADO also allows manipulation of database-defined data types as well as Binary Large Objects (BLOB), so you can store graphic images in a database and retrieve them as needed. Since Active Server components are Component Object Model (COM) objects, you can seamlessly distribute them over a network as out-of-process server components by using distributed COM (DCOM).

When you're ready to create an Active Server application, you can do so using the Microsoft Visual InterDev integrated development tool. In this article, we'll demonstrate how to create Active Server Pages to access and update a SQL database using ADO.

Calling All Support Staff

Let's create an Active Server Page that will display the names and telephone extensions of our support staff. As you may have guessed, we'll display this information directly from a Microsoft SQL database. In addition to the display, we'll provide editing functions so our staff can dynamically update the database by adding data to the Web page, editing information, or deleting data.

We assume that you already have ASP running on your NT server, and that you're using Microsoft SQL server release 6.5 with the update pack. We'll use Visual InterDev's built-in wizard to create the ADO and ActiveX Design Time control that displays the page to the user. When we're finished, our page will look like the one shown in **Figure A**.

{ewc mvimg, mvimage,!illust.bmp}
Figure A. Your users can dynamically add, edit, or delete SQL database data.

At the time of this writing, Visual InterDev is available in beta release. If you're a Site Builder Network member, you can download a free trial copy from **www.microsoft.com/sitebuilder**. Click here to connect to the sitebuilder Web page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Now let's quickly explore the basics of Visual InterDev — if you're already familiar with it, you can skip the next section and move directly to Database Basics.

This article includes the following topics:

Shamir Dasgupta is editor-in-chief of *Microsoft Web Builder*, The Cobb Group's journal for Web developers. His personal Website, www.xprssions.com, won three categories, including Grand Prize, in Microsoft's inaugural *Activate the Internet* competition in 1996.

Article ID: Q165293
Creation Date: 18-MAR-1997
Revision Date: 20-MAR-1997

The information in this article applies to:

® Microsoft Active Server Pages, version 1.0

Summary

In developing a Web Application, you may want to declare a table of data for use by one or more pages at application level scope. This article demonstrates how to declare, populate, and reference an array that has been declared at application level scope.

More Information

Repeat the following steps for a demonstration on referencing arrays with application scope.

1. On a computer running Active Server Pages, create a folder named ATEST and configure Internet Information Server (IIS) to recognize this directory as a virtual root with execute permissions.
2. Create a file in this directory named Global.asa. Copy and paste the following code into this file:

```
<SCRIPT LANGUAGE=VBSCRIPT RUNAT=SERVER>
SUB Application_OnStart
' This script executes when the first user comes to the site.
' or the global.asa is modified
' A simple fixed size array
Dim aFixed(3)
aFixed(1) = "Fixed"
aFixed(2) = "Size"
aFixed(3) = "Array"
' Cache the array as a member of the Application variables collection
Application("aFixed") = aFixed
' Declare a dynamic (resizable) array
Dim aColors()
' Allocate storage for the array
Redim aColors(16)
' Store values representing a simple color table
' to each of the elements
aColors(1) = "RED" ' [#FF0000]
aColors(2) = "GREEN" ' [#008000]
aColors(3) = "BLUE" ' [#0000FF]
aColors(4) = "AQUA" ' [#00FFFF]
aColors(5) = "BLACK" ' [#000000]
aColors(6) = "FUCHSIA" ' [#FF00FF]
aColors(7) = "GRAY" ' [#808080]
aColors(8) = "LIME" ' [#00FF00]
aColors(9) = "MAROON" ' [#800000]
aColors(10) = "NAVY" ' [#000080]
aColors(11) = "OLIVE" ' [#808000]
aColors(12) = "PURPLE" ' [#800080]
aColors(13) = "SILVER" ' [#C0C0C0]
aColors(14) = "TEAL" ' [#008080]
aColors(15) = "YELLOW" ' [#FFFF00]
aColors(16) = "WHITE" ' [#FFFFFF]
' Cache the array as a member of the Application variables collection
Application("aColors") = aColors
END SUB
</SCRIPT>
```

3. Create a file in the same directory named Color.asp. Copy and paste the following code into the file:

```
<%@ LANGUAGE="VBSCRIPT" %>
<HTML>
<BODY>
<H2>Arrays as Members of the Application variables collection.</H2>
<%
f IsArray(Application("aColors")) then
%>
   <H3>A Resizable Array</H3>
<%
   ' Put a reference to the array in a temporary variable
   ' for easy access
   aColors = Application("aColors")
   nColors = UBound(aColors)
%>
   <TABLE BGCOLOR=BLACK>
<%
      for i = 1 to nColors
         cColor = aColors(i)
       if cColor = "BLACK" then
          cBGColor = "WHITE"
       else
     cBGColor = "BLACK"
       end if
%>
         <TR>
            <TD BGCOLOR=<% =cBGColor %>>
               <FONT COLOR=<% =cColor %>><% =cColor %></FONT>
    </TR>
<%    next %>
   </TABLE>
<%
else
   Response.Write("Application('aColors') is not an array! <BR>")
end if
if IsArray(Application("aFixed")) then
%>
   <H3>A Fixed Size Array</H3>
<%
   aFixed = Application("aFixed")
   for i = 1 to UBound(aFixed)
       Response.Write(aFixed(i) & "<BR>")
   next
else
   Response.Write("Application('aFixed') is not an array! <BR>")
end if
%>
</BODY>
</HTML>
```

4. Save the files, launch a client browser such as Internet Explorer, and load the Active Server Page using a path similar to the following:

   ```
   http://<server>/Atest/Color.asp
   ```

   The browser should display the contents of the arrays, which were stored at the application level.

Observe that in the above sample it is only during application startup that the contents of the variables collection of the Application object is altered. For that reason, there is no need to lock the Application object. As with any other element of the variables collection of the application object, if the contents of the array were modified by another Active Server Page or via a session-level event, the Application object would need to be locked. For more information on the Lock and Unlock methods of the Application object, see the Active Server Pages Roadmap.

References

Active Server Pages Online documentation

Keywords : AXSFVBS kbprg kbhowto
Version : 1.0
Platform : NT WINDOWS

With the Publish to the Web Wizard, you can publish a set of Microsoft Access database objects to any combination of static HTML documents, IDC/HTX files, or Active Server Pages (ASP). By using the wizard, you can:

® Pick any combination of tables, queries, forms, or reports to save.

® Specify an HTML template to use for the selected objects.

® Select any combination of static HTML documents, IDC/HTX files, or Active Server Pages.

® Create a home page to tie together the Web pages you create.

® Specify the folder where you save your files.

® Use the Web Publishing Wizard to move the files created by the Publish to the Web Wizard to a Web server.

® Save the answers you provide the wizard as a Web publication profile, and then select that profile the next time you use the wizard. This saves you from having to answer the wizard's questions again.

To run the Publish to the Web Wizard, click **Save As HTML** on the **File** menu. For more information about using the Publish to the Web Wizard, search the Microsoft Access Help index for "Saving database objects, saving in Internet/Web formats."

```
'Error Handling Code Sample
Sub cmdSubmit_OnClick
    On Error Resume Next
    'Statement that might cause an error
    If Err <> 0 Then
        Msgbox "An error occurred. " & Err.Description
        Err.Clear
    End if
    'Statement that might cause an error
    If Err <> 0 Then
        Msgbox "An error occurred. " & Err.Description
        Err.Clear
    End if
End Sub
```

You can learn more about applications that use IDC/HTX files by reading about the Job Forum application. For information about the Job Forum application, see the Job Forum white paper, located at http://www.microsoft.com/accessdev/accwhite/jobforpa.htm.

For applications that require many users to access the database simultaneously, you should consider upsizing the Microsoft Access database back-end server to Microsoft SQL Server. For information about upsizing a Microsoft Access Web application to Microsoft SQL Server, see http://www.microsoft.com/accessdev/accwhite/upsizeweb.htm.

This document, from the Microsoft ActiveX SDK, explains the mechanism for licensing of ActiveX controls, so that control authors may sell design-time licenses to controls which allow web authors to distribute the controls on their pages for free run-time use.

See the License Package Authoring Tool in the Resources\Microsoft Tools section of the Library for information about the LPK_TOOL.

In order to create a market for ActiveX Controls to proliferate on the World Wide Web, it's essential to standardize a method for licensing ActiveX Controls embedded in HTML documents. Before discussing proposed licensing schemes, we establish criteria and objectives for such schemes.

One must first understand the existing **IClassFactory2** (ICF2) mechanism for licensing OLE Controls on the desktop-i.e. for use in Visual Basic. In the existing scheme, a VB programmer buys a design-time license for a control. When the programmer creates a VB form, the authoring tool embeds a free run-time license for the control inside the form, allowing anybody to use the form without paying for the control. The control itself participates in this process via the **IClassFactory2** interface, which exposes two important methods. The first method, RequestLicKey, is used by the authoring tool (VB) for querying the run-time license at design-time. The second method, CreateInstanceLic, is used by the run-time application (VBRUN.DLL) for instantiating the control using this run-time license. For more information on the existing scheme for licensing of Controls, see OLE Reference documentation for the **IClassFactory2** interface.

Having understood the desktop model for licensing controls, we specify objectives for online licensing of controls. The goals for online licensing make this model quite analogous to the desktop model:

- **Business model:** web authors purchase a design-time license to author HTML pages using a control. The run-time license for browsing the HTML page is typically free.
- **Security:** although a 100% foolproof mechanism would be great, it's not necessary. "Standard" online licensing need only be secure enough to prevent casual or unintentional copying of licensed controls. It's therefore not necessary to make piracy impossible-it's adequate to make piracy inconvenient. A control developer can always develop a private scheme that is more secure.
- **Convenience for authors:** licensing should not make things difficult for web authors. It's clear that the harder it is for web authors to use licensed controls, the less chance they will actually pay for such controls.
- **Compatibility:** the online model should accommodate existing controls that use the **IClassFactory2** licensing mechanism. This means each HTML page should have some associated control licensing information that is created at design-time and recognized by web browsers at run-time. In this model an HTML page is analogous to a Visual Basic form.
- **Versatility:** the online model must be versatile enough to support other online ActiveX Control containers, such as the NCompass ActiveX plug-in.

Online licensing of controls will continue to use the **IClassFactory2** mechanism that is implemented by existing licensed controls. An HTML page with licensed controls requires a single associated license package which stores the run-time licenses for all the controls used on the page. The license package (.LPK file) stores an array of (CLSID, license) tuples. The HTML page points to the license package via a relative URL reference inside the HTML. At design-time, authoring tools or utilities should use **IClassFactory2::RequestLicKey** to create the .LPK file and the associated HTML element (tag) that refers to it. At run-time, web browsers should extract the necessary run-time licenses from the license package in order to instantiate licensed controls embedded in the HTML page using **IClassFactory2::CreateInstanceLic**. In this model the HTML page is analogous to a Visual Basic form, HTML-authoring tools or helper utilities play the role of the VB Design Environment, and the web browser plays the role of the VB run-time (VBRUN.DLL).

It is important to note that the URL reference to the license package (.LPK) must be relative only. This makes it inconvenient to pirate licensed controls. Certainly it's less possible to accidentally pirate controls, because one can't copy an HTML page with a relative URL to a .LPK file and expect the page to still work. Since the URL to the .LPK file must be relative, in order to pirate a control one would have to copy the .LPK file knowingly. By using a plain-text format for the .LPK file, it is possible to include a legal copyright statement at the top of the file, dissuading anybody who downloads this file in order to copy it to a pirated site.

Web browsers can support this licensing scheme by posing additional obstacles for pirates who wish to copy .LPK files — for example making it difficult to download and save such files, or obscuring the cached copies of such files. Such obstacles do not make piracy impossible, but they make it even more inconvenient. Implementation of such obstacles is **optional** for web browsers, and is not a requirement of the proposed licensing scheme.

To justify the proposed scheme, we present a brief outline of the advantages, as well as a brief summary of alternatives that were ruled out for various reasons.

Advantages of the Scheme:

- **Convenience for web authors** — while a sophisticated authoring tool could make the design-time process completely invisible for web authors, even a simple design-time utility could make creation of .LPK files quite easy for authors that write HTML directly using tools such as Notepad.
- **Extra convenience for web sites** — a site with many web pages and many licensed controls could easily create one .LPK license package file containing all the licenses for all the pages on the site. After the one-time creation of this .LPK file, it is easy to refer to it from all pages on the site.
- **Inconvenience for pirates** — the average web developer cannot accidentally pirate licensed controls without realizing that they are breaking copyright laws. Although intentional piracy is not impossible, it is rather inconvenient and requires knowledge of the .LPK file format.

Other Licensing Alternatives that Were Considered

- **Tying the .LPK file to a particular site(s)** — one could achieve additional security by tying a license package to the site at which it would be used (for example, by including a list of URL prefixes or an X.509 certificate in the .LPK). However, the additional security is not worth the inconvenience to web authors, especially for publishers that host many sites, or for HTML-designer contractors that write HTML pages for many different publishers.
- **Tying the .LPK to a particular HTML page** — it is possible to use CRC digests to link a .LPK to a particular .HTML page. This was widely rejected as being too difficult at design time, and impossible for dynamically generated content.
- **Using digital certificates for foolproof security** — although in the future Microsoft intends to pursue 100% foolproof solutions to the licensing problem, for short-term needs (Internet Explorer 3.0) a simpler solution is much more practical.

HTML Syntax for Referencing .LPK Files

License packages must be inserted in HTML files using the existing <OBJECT> tag (element). The license package is interpreted by a *License Manager* object that is used to hand licenses to other Controls on the HTML page. The license package object must be embedded in the HTML page *before* any other objects that require licensing. The specific syntax for embedding license packages in HTML is as follows:

```
<OBJECT CLASSID = "clsid:5220cb21-c88d-11cf-b347-00aa00a28331">
     <PARAM NAME="LPKPath" VALUE="relative URL to .LPK file">
</OBJECT>
```

Note Internet Explorer 3.0 will only honor the *first* license package in an HTML page. If controls on this page require licenses that are not inlcuded in this license package, then they will afil to instantiate.

License Package File Format

The .LPK license package file is a plain text file, and must be labeled by servers with the MIME type *text/plain*. Any binary data in the file is *uuencoded*. This allows the file to be viewable by web browsers, so that anyone trying to copy the file would clearly notice the copyright statement at the top of the file. The contents of the file are as defined below. This file will be interpreted by a *License Manager* object.

.LPK Header	**This header identifies the file type: "*LPK License Package*"**
Copyright text or other legal statement	"Legalese" to dissuade casual copying of .LPKs.
LPK version GUID	In plain-text on a line by itself. This GUID is used to mark the beginning of the real license package data, and it is also used to identify the LPK file format version: "*{5220cb21-c88d-11cf-b347-00aa00a28331}*"

Uuencoded(Base64) license package:

```
struct {
    UUID uuidLPKFile;   // unique per LPK
    DWORD dwLicenses;   // number of licenses
in the file
    LICENSEPAIR aLicenses[]; // array of
license pairs
} LICENSEPACKAGE;

struct {
    CLSID clsid;        // clsid of object
    DWORD cchLic;       // Number of characters
in the license
    WCHAR ach[];        // License (saved as
UNICODE characters)
} LICENSEPAIR;
```

Authoring Tools

HTML authoring tools will make it easy to create HTML pages with embedded ActiveX Controls. Such tools should be responsible for creating the .LPK license package for licensed controls used on a page or on a web site. Since there may be a one-to-many mapping between LPKs and HTMs, this may be more difficult for pagebased authoring tools as opposed to web-based authoring tools (e.g. Microsoft FrontPage).

Clearly, tool support is necessary for Notepad HTML authors as well. The solution is a simple GUI tool that lists all controls that are installed on a machine with *design-time* licenses. The tool allows a user to create a .LPK license package by selecting which Controls should be included in the package. A second tool could parse HTML pages an create a .LPK file for all the controls that require licensing.

NCompass Support for Licensing ActiveX Controls

It is crucial that the licensing scheme described above works in other ActiveX-enabled web browsers, particularly in the NCompass ActiveX Plug-in. Because the plug-in specification only allows plug-ins to be specified using the <EMBED> syntax, therefore pages authored to work with both Internet Explorer 3.0 *and* NCompass will need to use syntax similar to the following:

```
<OBJECT CLASSID = "clsid:5220cb21-c88d-11cf-b347-00aa00a28331">
    <PARAM NAME="LPKPath" VALUE="relative URL to .LPK file">

    <EMBED SRC = "FOO.LPK">
</OBJECT>
```

1. Which of the following is an example of a business service?

{ewc mvimg, mvimage,!answer.bmp} A. An ActiveX control that displays a calendar on the screen.

{ewc mvimg, mvimage,!answer.bmp} B. A stored procedure in an SQL Server.

{ewc mvimg, mvimage,!answer.bmp} C. A database table that stores payroll information.

{ewc mvimg, mvimage,!answer.bmp} D. An ActiveX server component that can place an order in a database.

2. Which service provides the application with its graphical interface?

{ewc mvimg, mvimage,!answer.bmp} A. User services

{ewc mvimg.mvimage.!answer.bmp} B. Business services

{ewc mvimg.mvimage.!answer.bmp} C. Data services

3. Which member of a Web development team is responsible for running ActiveX server components?

{ewc mvimg.mvimage.!answer.bmp} A. Web Developer

{ewc mvimg.mvimage.!answer.bmp} B. Programmer

{ewc mvimg.mvimage.!answer.bmp} C. HTML Author

{ewc D. Graphic Artist

{mvimg,mvimage,!answer.bmp}

4. Which member of a Web development team is responsible for defining the architecture of a Web site?

{ewc mvimg,mvimage,!answer.bmp} A. Web Developer

{ewc mvimg,mvimage,!answer.bmp} B. Programmer

{ewc mvimg,mvimage,!answer.bmp} C. HTML Author

{ewc mvimg,mvimage,!answer.bmp} D. Graphic Artist

5. Which programming tool enables the programmer to create ActiveX server components and ActiveX controls?

{ewc mvimg.mvimage.!answer.bmp}

A. Visual InterDev

{ewc mvimg.mvimage.!answer.bmp}

B. Visual Basic

{ewc mvimg.mvimage.!answer.bmp}

C. Microsoft Transaction Server

{ewc mvimg.mvimage.!answer.bmp}

D. Microsoft SQL Server

6. Which tool provides transaction and resource management for ActiveX server components?

{ewc mvimg.mvimage.!answer.bmp}

A. Visual InterDev

{ewc

B. Visual Basic

{mvimg,mvimage,!answer.bmp}

C. Microsoft Transaction Server

{ewc mvimg,mvimage,!answer.bmp}

D. Microsoft SQL Server

{ewc mvimg,mvimage,!answer.bmp}

This article illustrates some ways in which Microsoft SQL Server can be used in conjunction with the Microsoft SQL Server services to provide powerful functionality to a client/server system. This article assumes knowledge of the SQL-Data Maintenance Objects (SQL-DMO) object model, Microsoft SQL Server services, and client/server development.

As this article is part of a larger series on developing Client/Server solutions, it is recommended that the following articles in the series be read as background information:

® The Architecture Process

® The Design Process

® The Basics

® Coding Guidelines

This white paper includes the following topics:

By Suzanne Adams
From *Active Server Developer's Journal*
Published by The Cobb Group

Traditional Web applications rely on external runtime engines to maintain state. However, Active Server applications must be able to share and maintain information across application pages. The Active Server, an extension of Microsoft Internet Information Server (IIS), facilitates this action by providing built-in Application and Session objects.

Active Server Session objects allow you to create, maintain, and abandon properties and instances of objects while controlling application workflow and object scope. In this article, we'll show you how to create and terminate Session objects. We'll also show you how to establish Session properties and include ActiveX objects in a Session object.

This article includes the following topics:

Susie Adams is a Senior Developer with Financial Dynamics, a Client-Server and Internet Solutions Consulting firm in McLean, Virginia. She specializes in Internet and Intranet web development and recently presented a conference on Visual InterDev at the Microsoft DevDays conference in Washington D.C.

By Suzanne Adams
From *Active Server Developer's Journal*
Published by The Cobb Group

Unlike most standalone applications, Web-based applications can have many potential points of entry. For example, a user could enter into a browser the name of virtually any page in your application. When the server receives the request, it tries to load that page — whether or not it's the page on which you want your users to begin. Fortunately, there's an easy way to avoid this potential problem.

Begin at the beginning

You can prevent users from bypassing the application login page by including in the GLOBAL.ASA file a script that handles the Session_OnStart event. In this article, we'll use the Response.Redirect method to ensure that all users start a session on a particular Web page. If the user requests a page other than the login page, our application automatically redirects the user to the correct login page.

When the Response or End method executes, the Active Server immediately stops processing the script in the GLOBAL.ASA file and in the application page that triggered the Session_OnStart event. Then, Active Server either ends the session or begins processing the new page.

It's important to note that any lines of script that follow the Redirect method call will not be executed. For this reason, you should always make the Redirect method the last call in your event script. **Listing A** shows the complete script for handling the Session_OnStart event.

Listing A. Directing users to the login page

```
<SCRIPT RUNAT=Server Language=VBScript>
Sub Session_OnStart
        'Check to see if the current ASP application page is the login page.
        StartPage = "/MyApp/Login.asp"

        'Retrieve the current page.
        currentPage = Request.ServerVariables("SCRIPT_NAME")

        'If the pages don't match, send the user to the correct login page.
if strcomp(currentPage, startpage, 1) then
    Response.Redirect( StartPage)
end if
End Sub
</SCRIPT>
```

Conclusion

Although you can't always control your users' actions, you may be able to prevent them from causing undue problems. In this article, we showed you how to redirect users to the login page of your Active Server application — regardless of which page they request.

Susie Adams is a Senior Developer with Financial Dynamics, a Client-Server and Internet Solutions Consulting firm in McLean, Virginia. She specializes in Internet and Intranet web development and recently presented a conference session on Visual InterDev at the Microsoft DevDays conference in Washington D.C.

By Suzanne Adams
From *Active Server Developer's Journal*
Published by The Cobb Group

ActiveX Data Objects (ADO) are objects that manage data integration in ASP files. By using ADO, you can rapidly serve custom information to users of your Active Server application.

You can use Session objects to help handle ActiveX Data Objects. Doing so lets you apply to ADO objects the same control you enjoy when you use Session objects. In this article, we'll show you how to integrate ADO with a Session object.

Sessions and ADO

When you use ADO, you manipulate data almost entirely with the Recordset object. To access data through a Recordset, you must first specify the data connection and the cursor for the Recordset. You can use Session objects to help manage these requirements.

One way you can apply the flexibility of Session objects to ADO-related tasks is to establish Session properties that govern the ADO processes in your scripts. Session objects allow you to control the scope and workflow of the ADO Recordset objects. To demonstrate, let's look at a sample Active Server application that combines ADO and Session objects.

Applying ADO

The script shown in **Listing A** creates Session properties that the application will use as connection object parameters. This script will reside on the GLOBAL.ASA page. You can also determine the scope of these objects at the time you create them.

Listing A. Creating Session properties

```
Sub Session_OnStart
'Project Data Connection
Session("DBConnNew_ConnectionString") = "DSN=maccess;DBQ=C:\maccess.mdb"
Session("DBConnNew_ConnectionTimeout") = 15
Session("DBConnNew_CommandTimeout") = 30
Session("DBConnNew_RuntimeUserName") = ""
Session("DBConnNew_RuntimePassword") = ""
End Sub
```

When a user starts your application, the Session_OnStart script establishes the connection object parameters. At this point, you can create a connection object from an ASP file in that application by using the script shown in **Listing B**.

Listing B. Creating a connection object

```
<%
If fNeedRecordset Then
    'Create connection.
Set DBConnNew = Server.CreateObject("ADODB.Connection")
DBConnNew.ConnectionTimeout = Session("DBConnNew_ConnectionTimeout")
DBConnNew.CommandTimeout = Session("DBConnNew_CommandTimeout")

'Open the connection.
DBConnNew.Open Session("DBConnNew_ConnectionString")

'Create the recordset.
Set rsRecordSet = Server.CreateObject("ADODB.Recordset")

'Create the command.
Set cmdTemp = Server.CreateObject("ADODB.Command")
```

```
cmdTemp.CommandText = "SELECT * from customer "
cmdTemp.CommandType = 1
Set cmdTemp.ActiveConnection = DBConnNew

'Open the recordset.
rsRecordSet.Open cmdTemp, , 1, 3
End If
%>
```

In this example, the Recordset we've created has page scope only. This means that only script located on the current page can access the Recordset object. On many occasions, however, you may need to access the same Recordset from another page.

To accomplish this expanded scope, you can turn Recordset objects into Session object properties that allow Session-scope access between application pages. The following script establishes the Recordset object rsRecordSet as a Session object property:

```
<%
If fNeedRecordset Then
Set Session("rs_Recordset") = rsRecordSet
End If
%>
```

After you establish the Session scope Recordset object, you can reference the Recordset from any page in your application. When the session ends, the object goes out of scope.

Another Example

Now, let's look at a more robust example. This sample application uses two ASP pages — EMPLOYEEDATA.ASP and SAVEDATA.ASP. The Active Server page DATA.ASP shown in **Listing C** populates our Recordset and allows the user to edit Recordset data.

Listing C. data.asp

```
<form method="POST" Action="savedata.asp">

<%
SQL="Select employee_id and employee_name and phone_number from employee "
Set DBConnNew = Server.CreateObject("ADODB.Connection")
Set cmdTemp = Server.CreateObject("ADODB.Command")
Set RS = Server.CreateObject("ADODB.Recordset")
cmdTemp.CommandText = SQL
cmdTemp.CommandType = 1
Set cmdTemp.ActiveConnection = Session("DBConnNew")

 'Create a recordset that has session scope.
Session("RS").Open cmdTemp, , 1, 3
%>
<input type="text" value=<%=Session("RS")("empoyee_id")%> name="employee_id" >
<input type="text" value=<%=Session("RS")("empoyee_name")%> name="employee_name" >
<input type="text" value=<%=Session("RS")("phone_number")%> name="phone_number" >

<input type="submit" name="BSave" value="  Save  ">
%>
```

For our example, we assume you have a database named *employee* that contains the fields *employee_id*, *employee_name*, and *phone_number*. When this page loads, it creates the Employee Data form, shown in **Figure A**. The Save Data button on DATA.ASP activates the SAVEDATA.ASP page and passes to it the three fields stored in the form's Recordset.

{ewc mvimg, mvimage,!illust.bmp}
Figure A. We'll use this form to enter data into our Active Server application.

The SAVEDATA.ASP file, shown in **Listing D**, saves the Recordset data to the employee database. The script also handles any errors and returns an error or success message, as appropriate.

Listing D. savedata.asp

```
<form method="POST">
<%
Session("RS")("employee_id")= Request("employee_id")
Session("RS")("employee_name") = Request("employee_name")
Session("RS")("phone_number") = Request("phone_number")
Session("RS").Update
If Err.Number <> 0 Then
Session("RS").CancelUpdate%>
<center> Employee Update ERROR! </center>
<%Else%>
    <center> Employee Update Successful! </center>
<%End If
%>
```

You can create the database connection for your Recordset object in the Session_OnStart event in the GLOBAL.ASA file instead of building a separate page in your Active Server application as we did in our example. However, we don't recommend that you do so. With ODBC 3.0 connection pooling, you have efficient connection handling without storing ADO connections at the session level.

Conclusion

Sessions offer a great way to manage ADO objects in an Active Server application. You can establish the scope of a Recordset object and implement its properties when you initiate the session.

Susie Adams is a Senior Developer with Financial Dynamics, a Client-Server and Internet Solutions Consulting firm in McLean, Virginia. She specializes in Internet and Intranet web development and recently presented a conference session on Visual InterDev at the Microsoft DevDays conference in Washington D.C.

If the WebBrowser control can’t display the full width or height of a Web page or document, it automatically displays scroll bars. However, in most cases, make the control wide enough to display the full width of a typical Web page so that users of your application don’t have to scroll horizontally.

In addition to being the fastest transaction processing database server in production today, Microsoft SQL Server can provide many other services. These range from the SQL-Data Maintenance Objects (SQL-DMO), which manipulate the database system through OLE Automation, to SQL Mail, which enables connectivity between your data and your MAPI-compliant mail system.

This article will introduce you to some of SQL Server's new services and show you how to get more mileage out of your current solutions and create new ones effectively.

There are many features of Microsoft SQL Server services that will make managing a client/server system easier. In some situations, the everyday processing of a system may be easier to create by building around these services and objects instead of providing custom processing. An example is using the Task objects to execute batch processors instead of the AT command line or similar schedulers. By using these objects instead of SQL or command line interfaces, you can use the functionality of the built-in Event Log to track success and failure, or you can use the pager and e-mail notification processes that are also built into Microsoft SQL Server. Neither of these services require the developer to provide much, if anything.

To make use of alerts, tasks, and the notification structure, you need an understanding of SQL Executive. SQL Executive is a service for Microsoft Windows NT that is provided along with Microsoft SQL Server. It has three basic jobs that offer lots of functionality:

® Process manager for Replication services

® Monitoring of Microsoft SQL Server status, providing alerts

® Process manager for Task services

SQL Executive is required for replication services, but we will be focusing on its other aspects. We can make use of SQL Executive because it is like a constantly cycling process running on the server. It is always interrogating the server and keeping track of the various aspects of both Microsoft SQL Server and the operating system. This aspect of its functionality can provide a lot of features with minimal effort.

For example, because of the cyclical nature of SQL Executive, you can have queries that are executed at any interval you want by using tasks. You don't have to program the interface or the "Keep Alive" smarts or anything! Just plug in the T-SQL or BAT or EXE and tell SQL Executive at what intervals to invoke the task. Voila! It happens all by itself. And when the job is executed, SQL Executive logs the start and stop times and all the errors and/or messages to the service log as well. This is great for those system operators who need remote processes to be run at certain times, but want tight control.

Tasks are fantastic for systems that have batch processing; alerts are great for systems that run unattended. Put together, these features can drastically improve the design even of simple systems by providing batch processing and call-back functionality while requiring very little effort from the system itself. For example, imagine a simple data entry system for expense reports. You have this great database that manages the expense reports for each of the employees in your department. It has a small, fast interface that looks something like the following:

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. Current transition interface for a theoretical system

Now you need to be able to interface your cool transaction system with the monstrosity they call the budget database. The goal is for each transaction to be validated against the current budget and for each of the items on the expense report to be validated and subtracted from the current budget, depending on the type of transaction and the budget that each item falls under. Pretty straightforward, right? The only problem you can see is that inserting each item from an expense report into the various categories can have serious implications for the rest of the budget and reporting processes. Because changing anything in the budget database has such big repercussions, the process for doing this type of validation and merging can take quite a while.

{ewc mvimg, mvimage,!illust.bmp}
Figure 2. Proposed transaction interface for a theoretical system

While this may seem like a normal situation, it unfortunately isn't one of the easier problems to solve. You could spend serious time trying to streamline the budget operations, you could try upgrading your hardware, you could try lots of things to make the transactions complete in less time. In the meantime, while you are working on getting the owners of the budget database to optimize their procedures so that you can release a usable product, the entire department must go without this much-needed functionality. Of course, after you spend so much time on it, there is no guarantee that any optimization you end up with will make the system usable. This is where SQL Services come in.

Instead of changing your current transaction model to facilitate extremely bulky transactions, you simply add two "hooks" to the current model. Hooks are like interfaces. They provide a means of talking about the linkages between systems that are not formally linked. You add one hook to the end of the current insert process. You might add the other hook as an OLE callback method or provide it in the form of the receipt of an e-mail message. The specific type of hook or callback is unimportant, and I'll go into more detail about this callback later. The key here is to design your system such that there are transactions into which hooks can be placed. From the previous illustrations, we can pick out two places for the initial hooks:

® pcUpd_ExpenseReportComplete

® pcUpd_ExpenseReportDeleted

In either case, the first step might be to store incoming and deleted data in a queue table in one database or the other. Which database the queue resides in will probably be driven only by which database has more free time. I'll make the assumption that the expense report database has more cycles to spare and put the queue and the task there. The second step is to invoke the task. Since tasks run asynchronously, the hook would be completed at that point and the client could continue while the task ran on the server.

The hook that we are inserting is probably a wrapper of some kind for many transactions that will move the data through the budget database. In most cases, all of this would probably be done in stored procedures to encapsulate the logic and make it run fast. It would be very simple to have the last step in the wrapper-stored procedure run a Microsoft SQL Server task. Doing that is trivial in SQL and would look like this:

```
sp_runtask 'TASK NAMED FOO'
```

As an alternative, the task could be kicked off from within the application as well. Working with the SQL-DMO is fairly simple, and this is how invoking the task might be implemented in Microsoft Visual Basic code:

```
Dim oSQLServer As New SQLOLE.SQLServer
Set oSQLServer = New SQLOLE.SQLServer
oSQLServer.Connect "MyServer", "MyLogin", "MyPassword"
oSQLServer.Executive.Tasks("Task Named Foo").Invoke
oSQLServer.Disconnect
oSQLServer.Close
```

Notice that even in this extreme scenario, the impact to the front-end code is minimal and straightforward. This is the main strength of a hook — it requires only a very small hole. The optimal case in this scenario requires no front-end changes at all, only changes to existing procedures.

As the task executes, SQL Executive will log start and stop times as well as any error conditions and, if required, will notify an operator through pagers, e-mail, or network messages. Upon completion, the task invoked in the first hook has many options for providing a callback to the client.

The callback hook can be implemented in any number of ways. The next section will lay out some alternative ways to implement a system-level callback in this context.

One option is that a callback is generated in the form of an e-mail message to the user with the results of the operations. Using Microsoft SQL Server Enterprise Manager, this tremendous feature is fairly easy to implement and could provide a great deal of information. Here is an example of how to send e-mail from within a Transact-SQL (T-SQL) statement:

```
xp_sendmail @recipients = 'sysadmin;acctreq',
                @copy_recipients = 'netops;msgops',
                 @subject = 'Task Completion Notification',
                @message = 'The task completed successfully.'
-- or to send a resultset would be
xp_sendmail @recipients = 'sysadmin',
                @query = 'select * from #logtable',
                 @attach_results = 'TRUE',
                 @subject = 'SQL Server Report',
                 @message = 'The task completed successfully. Log included.'
```

You can find information on the specifics of connecting your Microsoft SQL Server to your mail client on the MSDN Library CD.

A more sophisticated callback takes the form of a stored procedure that is invoked at the end of the first task. This procedure then manipulates an object on the server or uses Distributed COM (DCOM) to create a reference to the client (which at this point is acting as an automation server) and execute a callback method on the client, providing any information needed. This is what the process looks like:

{ewc mvimg, mvimage,!illust.bmp}
Figure 3. Process for executing callbacks on the client from the server

Here is what the extended procedure code might look like in the budget database:

```
DECLARE @object int
DECLARE @hr int
DECLARE @property varchar(255)
DECLARE @return varchar(255)
-- Create an instance of the callback server
EXEC @hr = sp_OACreate 'VBOLEServer.CallbackClass', @object OUT
IF @hr <> 0
BEGIN
    EXEC sp_displayoaerrorinfo @object, @hr
    RETURN
END
-- Set the WhoIsCalling property
EXEC @hr = sp_OASetProperty @object, 'WhoIsCalling', 'ThisProcedureName'
IF @hr <> 0
BEGIN
    EXEC sp_displayoaerrorinfo @object, @hr
    RETURN
END
-- Call the NotifyClient method
EXEC @hr = sp_OAMethod @object, 'NotifyClient', 32, 'Completed', @return OUT
IF @hr <> 0
BEGIN
    EXEC sp_displayoaerrorinfo @object, @hr
    RETURN
END
IF @return <> 0
BEGIN
    EXEC sp_SendAlert 'Failed To Notify Client. '
    RETURN
END
-- Destroy the server object
EXEC @hr = sp_OADestroy @object
IF @hr <> 0
BEGIN
    EXEC sp_displayoaerrorinfo @object, @hr
    RETURN
END
```

More information about manipulating OLE servers from Microsoft SQL Server can be found in SQL Server Books Online.

Still another option is to use the alert mechanisms to send notifications upon unsuccessful completion of the task. In many business processes, feedback should be provided only in the case of error conditions. It is for these types of situations that this option provides a cool alternative. Here is what the transaction process might look like on Microsoft SQL Server:

{ewc mvimg, mvimage,!illust.bmp}
Figure 4. Transaction process on the server

In order for the alert to fire, it must be created on the server beforehand. It can be created at run time using stored procedures or SQL-DMO code, but this isn't recommended. As well as creating the alert in advance, you need to assign criteria for the alert so that SQL Executive knows when to fire the alert. The simplest way to set up the criteria is to use a custom Error Number, which you also set up in advance.

There are two primary ways to actually get the alert to fire from inside the task. The first is to simply do a RAISERROR WITH LOG using the appropriate error number. The more appropriate way is to use the **xp_logevent** procedure. The SQL code for these two methods might look like this:

```
RAISERROR (@CUSTOM_MSG_NUMBER,0,1) WITH LOG
-- or the preferred syntax while within a task
EXEC xp_logevent @CUSTOM_MSG_NUMBER, @CUSTOM_MSG_TEXT, informational
```

For more information on using Extended Stored Procedures or alerts, see the documentation in SQL Server Books Online.

What all this boils down to is that the requirements for a callback are based entirely on the way you choose to run your business. If e-mail notifications are commonplace and considered an acceptable form of "receipt," then you might choose a limited code option. If your business operations include the use of alerts or must be tied closely to one another, then an OLE server or task-based solution might be the option for you. Development time and complexity are certainly issues to keep in mind as well.

At the end of the day, there are many ways to leverage the services of Microsoft SQL Server, depending on the level of control and timeliness required. With these services, it is possible to add a whole new spectrum of functionality to your working systems with a minimum of intrusion simply by changing your current model slightly. The main benefit is that the openness of these new services means that you are not required to jeopardize your current and viable solution simply to add new features.

The information in this article is a portion of the Visual InterDev Readmets.htm file, and explains how to install Transaction Server.

Setting Up

You install Microsoft Transaction Server on your computer by using the Setup program. The Setup program installs Transaction Server program files, sample applications, Help files, and other product components. Setup also installs the Java Virtual Machine that enables you to run components developed in Java.

Before You Run Setup

Before you install Microsoft Transaction Server, make sure that your computer meets the minimum requirements.

Hardware Requirements

To run Microsoft Transaction Server, you must meet certain hardware requirements, which include:

® Any Windows NT i386-compatible computer or Alpha AXP computer. For additional information regarding Alpha AXP computers, see "Alpha Platforms".

® A hard disk with a minimum of 30 megabytes available space for a full installation.

® A CD-ROM drive.

® Any display supported by Windows NT version 4.0.

® At least 32 megabytes of memory for i386-compatible computers.

® A mouse or other suitable pointing device.

Software Requirements

To run Microsoft Transaction Server, you must meet the following software requirements:

® Before you install Microsoft Transaction Server, you must install Microsoft Windows NT version 4.0 or later on your computer.

® If you want your components to access databases, use Microsoft SQL Server, version 6.5 or later, or another Microsoft Transaction Server compatible database.

® If you plan to create components with Microsoft Visual Basic, use Microsoft Visual Basic, Enterprise Edition, version 4.0 or later.

® If you plan to create components with Microsoft Visual C++ version 4.1, you must also install the Win32 SDK for Windows NT 4.0. If you are using Microsoft Visual C++ 4.2 or greater, you are not required to install the Win32 SDK. You can also use the Active Templates Library (ATL) version 1.1 or later.

® If you plan to create Internet applications, use Microsoft Internet Information Server version 2.0 or later and Microsoft Internet Explorer version 3.0 or later.

® If you want to use Microsoft Windows 95 clients with Microsoft Transaction Server, install DCOM for Windows 95. For the latest information on DCOM support for Windows 95, see http://www.microsoft.com/oledev on the World Wide Web.

In addition to these requirements, we strongly advise that you install the Windows NT 4.0 Service Pack 2 service pack on computers running the Transaction Server software. This service pack resolves several problems which you may encounter when running Transaction Server under heavy load. When available, you can pick up Service Pack 2 on http://www.microsoft.com/support.

We also strongly advise that you install the SQL Server Service Pack 2 on computers running the Transaction Server software. This service pack resolves several known problems. When available, you can pick up Service Pack 2 on http://www.microsoft.com/support.

Click here to connect to the Support Home page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Important If you reinstall Microsoft SQL Server after installing the Microsoft Transaction Server software, you

must install Transaction Server again. The Transaction Server software needs to always be installed after installing Microsoft SQL Server.

Setting Up Microsoft Transaction Server

Note This section is for participants in the Microsoft Transaction Server alpha or beta program.

If you participated in the Microsoft Transaction Server alpha or beta programs, you must uninstall those releases before installing this release. Do this as follows:

® Use your current Microsoft Transaction Server Explorer to delete all old packages.

® Use the Windows NT Explorer to delete any .pkg files.

® Use your current alpha or beta release Setup program to uninstall the Microsoft Transaction Server alpha or beta software.

To set up from compact disc

1. Insert the compact disc in the CD-ROM drive.
2. Run setup.exe, which is available in the root directory on the compact disc.
3. Follow the Setup instructions on the screen.

   When asked for a CD key, enter the Product ID contained on your CD. Please retain your Product ID because you will be asked for it if you contact Microsoft product support.

Validating Your Installation

You can use the Sample Bank application to validate your Microsoft Transaction Server installation. For more information, see "Validating Setup with a SQL Server Transactional Component" in the online Getting Started.

Validating Setup on Alpha Platforms

This release of Microsoft Transaction Server for Alpha platforms does not yet support Visual Basic or Java components. You can use the Hockey sample application to validate your installation (as explained in Getting Started), but you cannot use either the Visual Basic or the Java components of the Sample Bank application.

For instructions on how to validate your installation on Alpha platforms, see "Alpha Platforms" in the Visual InterDev Readmets.htm file.

Important Note Regarding SQL Server Setup If you install both SQL Server and Microsoft Transaction Server, you must install SQL Server first. If you later re-install SQL Server, you must then re-install Microsoft Transaction Server. This limitation will be eliminated in a future release of SQL Server.

Developers can take advantage of Active Server Session objects to address a number of programmatic issues that would otherwise be difficult or tedious to implement. The most common uses of Session objects include

® User identification information and user preferences

® Variables used routinely for calculations

® Application security

® Application data connection information

® Management of instances of ActiveX server objects

® Management of ADO (ActiveX Data Object) recordsets.

The greatest challenge Web developers face is maintaining application workflow and managing information throughout a user session. Active Server Session objects help workflow by providing the same consistency in Web-based applications that you expect in standalone applications. One of the keys to achieving smooth workflow is managing the scope of the variables and objects in your application.

The term *scope* identifies the extent to which a component instance, object, or variable is available within an application. The ActiveX Server maintains three levels of scope: *Application*, *Session*, and *Page*. **Figure A** illustrates the general scoping model of a typical Active Server application.

{ewc mvimg, mvimage,!illust.bmp}
Figure A. Objects in your Active Server application will have different levels of scope.

Application scope implies that information is available to all users of an application. Session scope narrows the focus to the individual user level. Page scope properties and objects are accessible only at parse time.

The ActiveX Server manages an Active Server application's workflow by separating a user session into three sections: the session's beginning, its unique identification, and its termination. An Active Server application consists of all the files located in its root directory, subdirectories, or virtual directories. The GLOBAL.ASA file, located in the root directory, is an optional file that allows you to trap Application and Session events.

The Application and Session objects have OnStart and OnEnd events. The Application_OnStart event occurs when any page within the application is requested. The Application_OnEnd event occurs when the server is shut down.

Session objects manage state information in the ASP application between multiple HTTP requests. The Session object persists within the application scope and is not destroyed or re-instantiated as the user navigates between pages. **Table A** shows the general guidelines for applying the Session object.

Table A. Guidelines for applying Session object

Syntax	Session.property|method
Properties	SessionID: This property returns the session identification for this user. Timeout: The timeout period for the session state for this application, in minutes.
Methods	Abandon: This method destroys a Session object and releases its resources.
Events	Session_OnStart Session_OnEnd

You use one of three methods to create Session objects. The first method occurs when a user requests an ASP file that is part of your application. If the GLOBAL.ASA file for that application includes a Session_OnStart procedure, your application will automatically create a Session object.

The second method occurs when a user references a variable stored as a session property. The third method occurs when a user references an object that has been defined using the <Object> tag in the GLOBAL.ASA file.

The Session_OnStart event fires when the server creates a new session. The server processes the Session_OnStart script before executing the requested page but after processing the Application_OnStart event. All of the built-in objects (Application, Request, Response, Server and Session) are available and can be referenced from the Session_OnStart event. Ordinarily, you'd place the Session_OnStart code on the GLOBAL.ASA page. The general syntax for handling the Session_OnStart event looks like this:

```
<SCRIPT LANGUAGE=ScriptLanguage RUNAT=Server>
Sub Session_OnStart
    Event code goes here
End Sub
</SCRIPT>
```

When the Active Server attempts to create a session, it first checks the HTTP header for a SessionID cookie. If it doesn't find a SessionID, the Active Server generates a Globally Unique ID (GUID) for the session and sends the GUID to the user's browser to store the SessionID in a cookie. This cookie is created by a complex algorithm to ensure that curious users can't compromise application security and is only available until the user's browser is closed.

Active Server manages sessions differently if the user's browser doesn't support cookies. The Active Server scans through the URLs on a page before sending text back to a user, determines which URLs reference another page of your application, and appends the SessionID as a URL parameter. The SessionID can then be passed from page to page in your application. Now that we've seen how to start a session, let's look at how you go about ending one.

As we've described above, the Active Server can automatically recognize when a session should begin; however, it has difficulty recognizing when a session should end. Unlike traditional standalone applications that terminate only when the user issues the Exit or Quit command, an Active Server application can terminate if the user merely navigates to a page not contained within the application. To further complicate matters, the user might want to return later to the application with the expectation that it maintained the session's state.

As a result, the Session_OnEnd event occurs only when a session calls the Abandon method or times out. Of the server objects, only the Application and Session objects are available in the Session_OnEnd event. The general syntax for the Sesson_OnEnd event, which resides in the GLOBAL.ASA file, looks like this:

```
<SCRIPT LANGUAGE=ScriptLanguage RUNAT=Server>
Sub Session_OnEnd
    Event code goes here
End Sub
</SCRIPT>
```

You can terminate a session only by allowing the Session Timeout property to expire or by explicitly calling the Session Abandon method. The Abandon method destroys all of the objects stored in a Session object and releases their resources. Let's take a look at these two properties in more detail.

Abandoning a Session

When you call the Session Abandon method, the server will process the information on that page and neatly terminate the session. Abandon is a great way to allow a user to end a session. For example, you could provide a Quit button on a form with the ACTION parameter set to the URL for an .ASP file that contains the command

```
<% Session.Abandon %>
```

When the server loads the page with the Abandon command, it automatically terminates the session and ends the application. Ending an application in this way gives you the opportunity to tie up any loose ends before releasing control to the user.

The Abandon method doesn't release the session state until all the script commands on the current page have been processed. For instance, the following script stores a value in the variable "MyDog" for the current session, even though it calls the Abandon method before assigning the variable:

```
<%
Session.Abandon
Session("MyDog") = "Willie"
%>
```

After the server processes the page, it destroys the variable and terminates the session. Now, let's look at the Timeout property.

Calling a Timeout

Unless you explicitly call the Abandon method, the session will automatically end after 20 minutes if the user doesn't request or refresh an ASP file in the application during that time period. If you want to set a timeout interval that's longer than the 20-minute default, you can set the value dynamically in the Session_OnStart Event on the GLOBAL.ASA page. For example, you'd use the following code to establish a Timeout value of 30 minutes:

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Session_OnStart
    Session.Timeout=30
End Sub
</SCRIPT>
```

Now, let's look at how to establish properties for a Session object.

You create a Session object property by storing a value to the Session object. For example, the following lines of script create Session properties that can be referenced from any page in your application:

```
<%Session("TheTime") = Now
Set Session("MyObj") = Server.CreateObject("Obj")%>
```

If you use Session properties several times on a page, you can store their values to local variables for better performance. Now, let's look at how you can incorporate ActiveX objects into a Session object.

Active Server applications can contain instances of Active Server components in Session and Application objects. To create an instance of an Active Server component, you can use the <OBJECT> tag in the Session_OnStart procedure in the GLOBAL.ASA file, or you can use the Server.CreateObject method.

If you use the <OBJECT> tag method, as we do in our first example, you must also include the RUNAT attribute and the SCOPE attribute. You set the RUNAT attribute to Server and the SCOPE attribute to Session or Application. You can reference the component by using either the registered name (PROGID) or the registered number (CLASSID) method.

The example shown in **Listing A** uses the registered name (PROGID) method to create an instance of the Ad Rotator Component. The script assigns the name MyAd to the component, which will remain in scope throughout the session. This script resides in the GLOBAL.ASA file.

Listing A. PROGID method

```
<SCRIPT RUNAT=Server Language=VBScript>
Sub Session_OnStart
        <OBJECT RUNAT=Server SCOPE=Session ID=MyAd PROGID="MSWC.Adrotator">
    </OBJECT>
End Sub
</SCRIPT>
```

When you declare a component using the <OBJECT> tag, the variable you assign to the component goes into the session or application namespace. This assignment means that you don't have to use the Session or Application object to access the component instance. For example, the following script line would open an instance of the Ad Rotator component:

```
<%  MyAd. GetAdvertisement("addata.txt") %>
```

Using CreateObject

You can also use the Server.CreateObject method to store an instance of a component in the Session object. The following example stores an instance of the same Ad Rotator object in a Session object:

```
<% Set Session("MyAdd") = Server.CreateObject("MSWC.Adrotator") %>
```

To display an advertisement on a page, you'd include the following code in an ASP file:

```
<% Set MyAd = Session("MyAdd") %>
<%= MyAd.GetAdvertisement("addata.txt") %>
```

Note that the Active Server doesn't instantiate a component that you declare with the <OBJECT> tag until a script in an ASP file references that component. However, the Server.CreateObject method instantiates the component immediately. For this reason, Session scope components you create using the <OBJECT> method offer better performance than do components you create using the Server.CreateObject method.

We've barely scratched the surface of what you can accomplish by using Session objects in your Active Server applications. Using Session objects is an extremely safe and efficient way of managing your application workflow in a multi-user environment such as the Internet. More important, Session objects work within the framework of the Active Server Pages model, allowing you to serve information in a mixed-client environment. For a more detailed example of how to use Session objects in a live application, check out Using Session Objects with ADO.

The Active Server Session object provides an invaluable set of tools to help manage state information for a single user over multiple HTTP requests. By using the concept of Session scope, your applications that consist of complex, interrelated pages and logic can now communicate more effectively between both active and static content application pages. The Active Server reduces development time by controlling workflow, using unique session identification, trapping Applications and Session events, and managing instances of Active Server objects.

The information in this article is a portion of the Visual InterDev Readmeis.htm file, and provides some late-breaking or other information that supplements the Microsoft Visual InterDev documentation. Late-breaking information about Visual Database Tools is provided in a companion Readme. For the most recent information about common questions and solutions to any issues you might have with Visual InterDev, see the FAQ on http://www.microsoft.com/vinterdev.

Click here to connect to the Visual InterDev Home page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Versions

This section lists the versions of all of the components in Visual InterDev:

Client

Internet Explorer: Version 3.01 (build 4.70.1215)

Visual InterDev: Version 1.0

Server

FrontPage Server Extensions: Version 2.0 (build 1112)

Active Server Pages: Version 1.0 (build 1.13.24)

Personal Web Server: Version 1.0a (build 1181a)

ODBC: Version 3.0 (build 2301)

Installation Notes

For information on installing Visual InterDev, see the *Getting Results with Visual Studio* book or the CD liner.

Removing Beta versions of Visual InterDev or Internet Studio components

If you have installed any pre-release versions of Microsoft Visual InterDev (formerly known as Internet Studio) or any of the components that are included with Visual InterDev such as the FrontPage Server Extensions or Active Server Pages, you must remove them. This will help to ensure that the released versions of these components will be correctly installed.

Note For the most recent updates to these instructions, please visit the Visual InterDev Web site at http://www.microsoft.com/vinterdev (see above).

---

u **To remove all Beta versions of Visual InterDev or Internet Studio**

1. Double-click the Add/Remove Programs icon in Control Panel, and then remove the Beta client software, labeled "Microsoft Internet Studio" or "Microsoft Visual InterDev".
2. Restart your computer.
3. Remove all the Beta components in the same way. This includes Microsoft Active Server Pages, Microsoft FrontPage 97 Server Extensions, and a beta-only component, the Microsoft Internet Studio Server.
4. After the last item has been removed, or when asked, restart your computer.
5. Find your Beta software directory, either:

   X:\Program Files\DevStudio\Istudio

   -or-

   X:\Program Files\DevStudio\VInterDev

   where X is the hard drive where you have Windows installed.
6. Now that you have uninstalled the Beta software, back up any data files in these directories and then delete the Beta software directories. You can remove these Beta directories only if you have already uninstalled the Internet Studio or Visual InterDev Beta using the Add/Remove Programs icon in Control Panel.

7. Remove several Beta registry keys from your system that were added by the Beta.

Note If you prefer an automated method of updating the registry, you can download a program from http://www.microsoft.com/vinterdev that will safely remove these keys for you. (See above.)

Be very careful when deleting registry entries: deleting the wrong entries can prevent Windows from restarting. If necessary, you can restore the registry by following the instructions in the "Restoring the Registry" section of the Registry Editor Help system. Before rebooting, be sure to read these instructions in case the system is not restored correctly and does not respond.

u **To remove the registry keys from the Beta**

a. Click the Start button and choose Run.

b. Type regedit.

c. Navigate in the left pane to each of the following locations, select the specified folder in the left pane, and then press Delete. If you can't find a folder, skip it. For example, to delete the first key, navigate to

HKEY_CURRENT_USER\Software\Microsoft\Devstudio\5.0\Html

Then, click the HTML folder, and press Delete. Note Do not delete any of the preceding folders, such as "5.0" or "Devstudio".

Delete the following registry folders:

HKEY_CURRENT_USER\Software\Microsoft\Devstudio\5.0\Html

HKEY_CURRENT_USER\Software\Microsoft\DevStudio\5.0\IstudioProject

HKEY_CURRENT_USER\Software\Microsoft\FrontPage (Internet Studio Edition)

HKEY_CURRENT_USER\Software\Microsoft\FrontPage (Visual InterDev Edition)

HKEY_CURRENT_USER\Software\Microsoft\FrontPage (Visual Studio Edition)

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\DevStudio\5.0\Products\Microsoft Visual InterDev

HKEY_LOCAL_MACHINE\Software\Microsoft\DevStudio\5.0\Products\Microsoft Internet Studio

8. Close the registry editor.

9. Continue by installing the released version of Visual InterDev.

If you still have problems with the server pieces on Windows NT and you have had a Beta version on your computer

Many install issues can be traced to files with improper version stamps, which can be a by-product of Beta versions. Improper version stamps can prevent a file from getting uninstalled and overwritten when reinstalling, leaving your computer in a less-than-optimal or even a non-working state. In almost every case, uninstalling the Microsoft Internet Information Server and then Internet Explorer will reduce your computer to a generic state from which to rebuild. Previous Beta testers should consider the steps below only if the above steps do not work and you are familiar with the Windows NT operating system.

® Use the Add/Remove Programs icon in the Control Panel:

1. Uninstall Microsoft Internet Information Server.
2. Uninstall Microsoft Internet Explorer.
3. Uninstall Microsoft FrontPage.
4. Uninstall any Visual Studio (DevStudio) client programs (including Visual InterDev)

® Delete the Bin directories manually:

Note You should attempt to delete files and directories manually only after you try to uninstall using the Add/Remove programs icon in the Control Panel.

The Bin directories are usually installed in the \Program Files directory as Microsoft FrontPage\Bin, Microsoft

FrontPage\Servsupp, and DevStudio. Back up any personal data files in these directories.

® After deleting the Bin directories:

1. Reinstall Microsoft Internet Information Server.
2. Reinstall Microsoft Internet Explorer 3.01 or later.
3. Reinstall all the Visual InterDev server pieces again.
4. Reinstall the client pieces if the client will be run on the same computer as the server components.

Installing Windows NT Workstation Peer Web Services After Installing Visual InterDev

This installation order will install incorrect ODBC files. To correct this problem, rerun the Active Server Pages item in the Visual InterDev Master Setup dialog box.

Installing the Oracle ODBC driver

Microsoft Visual InterDev includes a new, recommended Oracle ODBC driver written by Microsoft that you can install for use with Microsoft Visual InterDev. The Oracle ODBC driver is not included with the typical install.

u **To install the Oracle ODBC driver:**

1. In the Master Setup dialog box, choose Custom.
2. Choose ODBC.
3. Select the Oracle driver.

-or-

1. Insert the Microsoft Visual InterDev CD-ROM in your CD-ROM drive, and go to the \Server\Oracle directory.
2. Launch the Setup program in this directory.

Note This driver should be installed on any Web server computers from which you want to access Oracle databases. Note also that the Oracle driver requires SQL*NET 2.3 or later to run properly. You can obtain this software from Oracle Corporation.

Installing the SQL Server Service Pack 2

If you plan to use Visual InterDev with SQL Server 6.5, you will need to first install Service Pack 2 in order to fix some problems that prevent the Visual InterDev database tools from running properly. Service Pack 2 is included on the Visual InterDev CD ROM.

u **To install the SQL Server Service Pack 2:**

1. Insert the Microsoft Visual InterDev CD-ROM into the CD-ROM drive on the computer that is running SQL Server 6.5, and go to the \Server\SQLSrvSp directory.
2. Launch Setup.exe.

RISC versions of Visual InterDev Server components are available

Dec Alpha and Power PC versions of the FrontPage Extensions are available on: http://www.Microsoft.com/FrontPage. Click here to connect to the FrontPage Web page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Dec Alpha and Power PC versions of the Active Server Pages component are available on: http://www.Microsoft.com/IIS. Click here to connect to the Internet Information Server Web page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Upgrading from FrontPage Personal Web Servers

If you installed Microsoft FrontPage on Windows 95, you might have installed the Front Page Personal Web Server instead of the Microsoft Personal Web Server. In order to use Active Server Pages with Visual InterDev, you need to install the Microsoft Personal Web Server that ships with Visual InterDev.

If you attempt to install the Microsoft Personal Web Server that comes with Visual InterDev on a computer that already has the Front Page Personal Web Server, you can get the following error when you click Personal Web Server on the Visual InterDev Master Setup dialog box:

> "An older version of Microsoft Personal Web Server was found on your machine. You must uninstall the existing version before setup can continue."

If you have this configuration on your computer, see http://www.microsoft.com/frontpage/upgrade/engupgrade.htm for step-by-step instructions on how to migrate your content to the new Microsoft Personal Web Server for Windows 95, or for information on how to make both Web servers co-exist on the same system.

This article is a portion of the Active Server Pages Readme file, and discusses issues with which you should be familiar prior to installing ASP.

Minimum Disk Space

You should have a minimum of 30 megabytes of free disk space in order to install ASP.

Microsoft Windows NT and Microsoft Windows 95 Web Servers

Before you install ASP on a computer running Microsoft Windows NT or Microsoft Windows 95, you must have already installed one of the following Microsoft Web servers:

® On Windows NT Server, Internet Information Server version 2.0

® On Windows NT Workstation, Peer Web Services version 2.0

® On Windows 95, Personal Web Server version 1.0a

Stop Applications and Services That Use ODBC Prior To Installing ASP

Before you install ASP, be sure to stop all applications and services that might use ODBC, including Microsoft SQL Server and Microsoft Access.

Installing ASP on a Windows NT Backup Domain Controller

If you installed Microsoft Internet Information Server as part of installing Microsoft Windows NT Server, and you made your computer a Backup Domain Controller during Windows NT setup, you will receive an error message when you install ASP. To successfully install ASP under these conditions:

1. Select Active Server Pages Uninstall from the Microsoft Internet Server (Common) Program Group in order to remove ASP.
2. Reinstall IIS.
3. Reinstall ASP.

Installing Microsoft VM for Java

ASP supports server components written in Java. To run Java server components, you must first install Microsoft VM for Java on your Web server. VM is available for download from www.microsoft.com, from the following locations:

® i386 (for Windows NT and Windows 95)

http://www.microsoft.com/java/jvmi386.cab

® Alpha (for Windows NT)

http://www.microsoft.com/java/jvmalpha.cab

® PPC (for Windows NT)

http://www.microsoft.com/java/jvmppc.cab

VM is also included in Windows NT version 4.0 Service Pack 2 and later.

Database Access Component Include Files

If you plan to use the ADO constants defined in the include files Adovbs.inc and Adojavas.inc (for VBScript and JScript, respectively), be sure to install the Adventure Works sample site. These files are located in the Aspsamp\Samples directory, which is installed in the \Inetpub directory by default.

When you start Visual InterDev, it launches Microsoft Developer Studio's Integrated Development Environment (IDE). If you're a C++ or J++ programmer, you'll find this environment familiar. The left pane displays the project and its associated files, and the right pane displays the source code. Visual InterDev includes both a source HTML editor and a WYSIWYG (what you see is what you get) version. In addition, it includes specialized editors for ActiveX Controls, Java applets, and plug-ins.

Visual InterDev's WYSIWYG editor is really a version of Microsoft FrontPage 2.0, which lets you easily insert ActiveX Controls, Java applets, and client-side scripts into your application. The source editor is a new-generation smart editor that uses color syntax to highlight functionality. If you're a J++ programmer, you can bring up the Java development environment within a Visual InterDev project.

Visual InterDev also includes extensive multimedia support and a very useful tool called Microsoft Music Producer, which you can use to generate MIDI music files on the fly and embed them on your Web pages.

To find out more about Visual InterDev's many features, click here to visit the Visual InterDev page on the Microsoft Web site.
{ewc mvimg, mvimage,!intjump.bmp}

For this example, we've created a simple SQL database named *Employees*. Our database contains a table that includes just three fields: *First Name*, *Last Name*, and *Extension*. To follow along with our example, you may want to build an *Employees* database of your own. Note that you don't always have to use a prebuilt database and table — Visual InterDev offers a feature that lets you create a database on the fly.

After you start Visual InterDev, create a Web using the Web Project Wizard. To do so, select New from the File menu. In the resulting dialog box, select Web Project Wizard from the Projects tab, as shown in **Figure B**, and type *test* in the Project Name input box. Click OK to continue to the next step.

{ewc mvimg, mvimage,!illust.bmp}
Figure B. Use the wizard to create a new Web project.

In the next dialog box, select the Web server where the new Web will reside and click Next. Remember that in order to run Active Server Pages, your Web server must be running on Windows NT 4.0 and IIS 3.0. In the following dialog box, specify a name for your Web, as shown in **Figure C**. To minimize confusion, it's wise to use the same name for both the project and the Web. Click Finish to build the Web.

{ewc mvimg, mvimage,!illust.bmp}
Figure C. Give your new Web a name.

When you create a Web using Visual InterDev, the program automatically creates a GLOBAL.ASA file. This file, which is loaded into memory when any user views a page for the first time, contains blank *Event Procedure* stubs. You can add script to these stubs — the scripts will execute when the application or session starts or ends.

After you add a data connection to a Web project, Visual InterDev adds the script that stores connection information in session variables in the *Session_OnStart* event procedure. The Visual Database Tools use this information to connect to the database server at design time and the design-time ActiveX Controls to connect to the database at runtime.

Listing A shows a typical GLOBAL.ASA file. We'll revisit this file after we add our database connector. Now that we've created our Web, we can add the data source and build our project.

Listing A. A bare-bones GLOBAL.ASA file

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">

'You can add special event handlers in this file that will
'run automatically when special Active Server Pages
'events occur. To create these handlers, create a
'subroutine with a name from the list below that
'corresponds to the event you want to use. For example,
'to create an event handler for Session_OnStart, you'd
'put the following code into this file (without comments):

'Sub Session_OnStart
'**Put your code here **
'End Sub

'EventName               Description
'Session_OnStart      Runs the first time a user runs any
'page in your application
'Session_OnEnd       Runs when a user's session times
'out or quits your application
'Application_OnStart  Runs once when any user runs the
'first page of your application for the first time
'Application_OnEnd   Runs once when the web server shuts down

</SCRIPT>
```

If you think Visual InterDev is full of wizards, you're right — and we'll use one of them to add a data source to our project. Select New from the File menu and highlight Data File Wizard on the File Wizards tab. Enter a filename in the File Name input box and click OK. The wizard will launch a seven-step process in which you'll provide information on the database source, drivers, tables, and user functions.

Step 1

First, you'll click New to specify a new data source name. In the resulting dialog box, click the New button that's beside the DSN Name input box. The next screen lists all available ODBC data sources.

Highlight SQL Server, as we've done in **Figure D**, and click Next to continue. In the next screen, you'll specify a name for the data source. For this example, enter *aspdj* and click Next. The dialog box will now display a summary of your selections; click Finish to complete the process.

{ewc mvimg, mvimage,!illust.bmp}
Figure D. Visual InterDev displays all available ODBC data sources in your systems.

At this point, you'll need to specify the SQL Server name and login information. Click the Options button to specify that you want to use the SQL database *Employees*, as shown in **Figure E**.

{ewc mvimg, mvimage,!illust.bmp}
Figure E. You'll need to provide security information and specify a SQL database.

Click OK to return to the main Select Data Source dialog box — our new Data Source Name (DSN) now appears on the list. Highlight the newly created DSN or type *aspdj.dsn* in the DSN Name input box, then click OK. Visual InterDev will display the SQL server login dialog box once again to verify the connection setting; click OK to accept the values. The Data Form Wizard will return to the original dialog box. Enter the title *The Cobb Group Demo* in the corresponding input box and click Next to move to step 2.

Step 2

In this step, you'll choose a record source. Simply highlight Table and click Next.

Step 3

The resulting dialog box will list the table's fields. You can now select which fields you want to display — in this case, First Name, Last Name, and Extension.

When we created our table, we had to give our fields names that didn't contain spaces: firstname, lastname, and extension. However, we want to display our field headings as readable headings. To do so, select the advanced options as shown in **Figure F** and enter the new heading in the Alternative Field Label input box. For our fields, we've used *First Name*, *Last Name*, and *Extension,* respectively. Click Next to continue.

{ewc mvimg, mvimage,!illust.bmp}
Figure F. You can specify the fields you want to display by moving them from left to right.

Step 4

In this section of the wizard, you can customize the edit options for your table so that it will have only the functionality you want for your users. In our case, we've left all the radio buttons and check boxes selected. Click Next to go to step 5.

Step 5

In this step, you can specify the view option for your table. Visual InterDev gives you considerable flexibility in how you view your data. For example, you can limit the number of records you display on a page so your viewers won't have to scroll to see a specific record. Accept the default value (10) in this screen and click Next to continue.

Step 6

Visual InterDev includes built-in style sheets you can apply to your page. Select a design scheme and click Next.

Step 7

The final step doesn’t require you to do anything except click Finish to accept the values you’ve entered in the previous steps. If you want to go back and modify some values, this is the time to do so. When you’re ready, click Finish and sit back — the wizard will do all the hard work.

After the wizard finishes creating the necessary files, examine the File View in the left pane. You'll notice that Visual InterDev has added three files; it also modified the GLOBAL.ASA file. Remember the contents of the generic GLOBAL.ASA file in **Listing A**, in Database Basics? **Listing B**, in the Conclusion, shows the same file after modification — notice the data connection section that Visual InterDev added.

Visual InterDev also adds the three files Form, List, and Action. The data form project name forms the prefix of the actual filenames. For example, if your project name is Test, Visual InterDev will name these files *TestForm.asp*, *TestList.asp*, and *TestAction.asp*. Each of these pages contains the necessary Visual Basic scripts and functions and Design Time Controls, all generated by Visual InterDev. You can view the source codes by double-clicking on the appropriate ASP file in the left pane, as shown in **Figure G**.

{ewc mvimg, mvimage,!illust.bmp}
Figure G. You can view, add to, or modify the source code from Visual InterDev development environment.

Preview the *TestForm* page by right-clicking on the file and choosing the preview option. Visual InterDev includes a built-in browser; however, you can view your application in any browser that's installed on your system. We previewed the file *TestForm.asp* in Internet Explorer, as shown in **Figure A**, in the first section, Adding Database Functions with ASP.

Visual InterDev provides a powerful integrated environment for rapid development. By following our example, you can create robust database-driven applications without writing a single line of code.

Listing B. Modified GLOBAL.ASA file

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">

'You can add special event handlers in this file that will
'run automatically when special Active Server Pages
'events occur. To create these handlers, just create a
'subroutine with a name from the list below that
'corresponds to the event you want to use. For example,
'to create an event handler for Session_OnStart, you'd
'put the following code into this file (without comments):

'Sub Session_OnStart
'**Put your code here **
'End Sub

'EventName                Description
'Session_OnStart      Runs the first time a user runs any
'page in your application
'Session_OnEnd       Runs when a user's session times
'out or quits your application
'Application_OnStart  Runs once when any user runs the
'first page of your application for the first time
'Application_OnEnd   Runs once when the Web server shuts down

</SCRIPT>
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Session_OnStart
    '==Visual InterDev Generated - DataConnection startspan==
    '--Project Data Connection
    Session("Employees_ConnectionString") =
"DRIVER={SQLServer};SERVER=BORG;UID=sa;PWD=; APP=Microsoft (R) Developer
Studio;WSID=DREAMER; LANGUAGE=us_english;DATABASE=Employees"
        Session("Employees_ConnectionTimeout") = 15
    Session("Employees_CommandTimeout") = 30
    Session("Employees_RuntimeUserName") = "sa"
    Session("Employees_RuntimePassword") = ""
    '==Visual InterDev Generated - DataConnection endspan==
End Sub
</SCRIPT>
```

You can add more ActiveX controls to the Toolbox by right-clicking the **Controls** tab on the Toolbox, and then clicking **Additional Controls**. Select the controls you want to add, and then click OK.

Click here to launch the ActiveX Version Utility Setup program.
{ewc mvimg, mvimage,!exec.bmp}

The ActiveX Version Utility is a utility that builds a 4-part version number for ActiveX controls and DLLs. It also creates HTML <OBJECT> tags for non-ActiveX DLLs.

After running Setup, run HTMLverAndCLSID from the Start menu to run the ActiveX Version Utility.

u **To get the version number for an ActiveX control**

1. Run **HTMLverAndCLSID** from the Start button.
2. Click **Locate one file and get its version number in n,n,n,n required format**.
3. In the Locate dialog box, click the .ocx, .exe, or .dll file you want to retrieve the version number from, and click **Open**.
4. Click **Copy** to copy the version number to the clipboard.
5. In the <OBJECT> tag for the control in the Web page, paste the version number into the CODEBASE attribute.

Note This tool is in the \Tools\ActX_Ver directory on the *Mastering Web Site Development CD-ROM*.

Script is added to the global.asa file to define the **Session_OnStart** event procedure and to store data connection information in session variables.

For more information, see Viewing the Global.asa File.

Controls are not added to pages. Script is added to the global.asa file.

For more information, see Viewing the Global.asa File.

Script is added only to the global.asa file.

For more information, see Viewing the Global.asa File.

Script is added to the global.asa file.

For more information, see Viewing the Global.asa File.

The Data Form Wizard generates more than just one .asp file. It can generate the .asp files form, list, and action, depending on the options you selected when you ran the wizard.

For more information, see Viewing Data Form Files.

The Data Form Wizard can generate the .asp files form, list, and action, depending on the options you selected when you ran the wizard.

For more information, see Viewing Data Form Files.

The Data Form Wizard generates .asp files, not .htm files.

For more information, see Viewing Data Form Files.

The Keyset and Static cursor types support paging.

For more information, see Inserting the Data Range Header.

The Dynamic cursor does not support paging.

The Keyset and Static cursor types support paging.

For more information, see Inserting the Data Range Header.

The Keyset and Static cursor types support paging.

For more information, see Inserting the Data Range Header.

The Dynamic and Forward Only cursor types do not support paging.

The Keyset and Static cursor types support paging.

For more information, see Inserting the Data Range Header.

Both ActiveX controls and Java applets can have a visual interface.

For more information, see ActiveX Controls vs. Java Applets.

The safety level set in Microsoft Internet Explorer determines whether unsigned ActiveX controls will be available to users. Java applets are not affected by the safety level.

For more information, see ActiveX Controls vs. Java Applets.

ActiveX controls are self-contained applications that can be run after they have been installed and registered on the user's computer. Java applets are run by a Java Virtual Machine.

For more information, see ActiveX Controls vs. Java Applets.

ActiveX controls are self-contained applications that are not interpreted. Java applets are run by a Java Virtual Machine.

For more information, see ActiveX Controls vs. Java Applets.

HTML Layouts provide an easy interface for placing controls in a two-dimensional format.

For more information, see Using HTML Layouts.

An HTML Layout is a file that includes the tags for ActiveX controls. The file can be included in multiple Web pages, but it is not a template file.

For more information, see Using HTML Layouts.

You can place only ActiveX controls in an HTML Layout.

For more information, see Using HTML Layouts.

HTML Layouts contain ActiveX controls. They do not contain forms.

For more information, see Using HTML Layouts.

The CODEBASE attribute is required only if the control has not been installed on the user's computer.

For more information, see The <OBJECT> Tag.

The ID attribute is required only if the control will be used in a script.

For more information, see The <OBJECT> Tag.

The CLASSID attribute identifies the control to be used.

For more information, see The <OBJECT> Tag.

If you do not set the WIDTH attribute, the control will use internal properties after it is instantiated to determine its size.

For more information, see The <OBJECT> Tag.

The CODE, WIDTH, and HEIGHT attributes are required in the <APPLET> tag.

For more information, see The <APPLET> Tag.

The WIDTH and HEIGHT attributes are required in the <APPLET> tag.

For more information, see The <APPLET> Tag.

The NAME attribute is required only if you are using an applet in client-side script.

For more information, see The <APPLET> Tag.

The WIDTH and HEIGHT attributes are required in the <APPLET> tag, but the NAME attribute is required only if you are using the applet in client-side script.

For more information, see The <APPLET> Tag.

You use the ARCHIVE attribute with the <APPLET> tag to specify a .zip file archive.

For more information, see Setting the CODEBASE Attribute.

VALUE is not a valid attribute in the <OBJECT> tag.

For more information, see Setting the CODEBASE Attribute.

The CODETYPE attribute of the <OBJECT> tag specifies the media type for the code.

For more information, see Setting the CODEBASE Attribute.

The CODEBASE attribute of the <OBJECT> tag can reference an .ocx file, a .cab file, or an .inf file.

For more information, see Setting the CODEBASE Attribute.

You use the ARCHIVE attribute with the <APPLET> tag to specify a .zip file archive for a Java applet.

For more information, see Distributing CLASS Files.

To distribute a Java applet in a .cab file, you create a <PARAM> tag and set the NAME attribute to cabbase and the VALUE attribute to the name of the .cab file.

For more information, see Distributing CLASS Files.

You use the CODEBASE attribute with the <OBJECT> tag to specify a .cab file for an ActiveX control.

For more information, see Distributing CLASS Files.

To distribute a Java applet in a .cab file, you add a <PARAM> tag with the NAME attribute set to cabbase, not to codebase.

For more information, see Distributing CLASS Files.

Signing a control guarantees only that the control has not been changed since it was signed by the author, and not that it is safe to use.

For more information, see Code Signing.

A digital signature identifies the author of the control.

For more information, see Code Signing.

Marking a control (not signing it) specifies that the control is safe to be used in a script.

For more information, see Code Signing.

A digital signature verifies only that the author of the control is who she says she is. It does not imply any type of licensing.

For more information, see Code Signing.

The control is named ValidateOrder, and the name of the event is OnClick. When you name a procedure ValidateOrder_OnClick, it runs automatically when the user clicks the control.

For more information, see Writing Event Procedures.

The name of the event procedure must include the name of the control that is set with the NAME attribute of the <INPUT> tag.

For more information, see Writing Event Procedures.

This article illustrates some fundamental concepts that should be considered when designing client/server solutions in Visual Basic. It is part of a series beginning with the article The Architecture Process, also available in the Microsoft Developer Network (MSDN) Library.

This white paper includes the following sections:

This technical article illustrates some fundamental concepts that should be considered when architecting client/server solutions in Visual Basic. It is geared toward developers who are providing client/server business solutions using the Visual Basic development environment. The concepts in this article are particularly useful for developers with an understanding of transaction processing systems.

This white paper includes the following topics:

This technical article illustrates and discusses several fundamental coding practices and guidelines recommended for developing client/server solutions in Visual Basic. It is part of a series beginning with the article Client/Server Solutions: The Architecture Process, also available in the Microsoft Developer Network (MSDN) Library.

Overview

The following are practices and techniques that should be used when writing Visual Basic code. Most of the items do not deal with style, but with proven techniques for writing solid code. The more habitual that good coding practices become for you, the easier it will be to read and modify your code. Standard techniques and practices also help in debugging and sharing code among programmers. Using these standards should provide some commonality among programming styles that otherwise might differ greatly.

You can modify these ideas to fit your individual coding style and the stability level of the project. Common sense should dictate which suggestions should be rules for code review and which are optional guidelines.

Table 1. Default Settings

Item	Comment
Option Explicit	This is pretty standard. Always put this in every module and every form.
Option Compare	String comparisons are handled differently depending on how this is set; therefore, always explicitly set this value in modules and forms.
Option Base	This changes what the default lower bound of arrays is set to. Always explicitly set this value in modules and forms.
DefVar	This will declare variables without type definitions to be a particular type. Always explicitly declare all variables to be of a certain type. Also define the defaults.
Text	Always save all files as text. This will make available the many tools that utilize Visual Basic code stored as text. Examples are analyzers, source control systems, etc. Also in the case of reconstruction or reuse, text files are easier to manipulate than binary data files.

Table 2. External Declarations

Item	Comment
Declaration organization	External declarations should be grouped consistently and all should be included in the same module. Any grouping convention may be used as long as consistency is maintained. Some common groupings would be by library or by task type.
Pointers to strings	Pointers to strings are common when using API functions. Because Visual Basic handles strings differently and because Visual Basic lacks pointers, you must handle these situations specially. Use the **STRCPY** API function to copy from a long pointer to a string. When copying a pointer to other data types, including user-defined data types (UDTs), use the **MEMCPY** API function.

Table 3. Operations

Item	Comment
Proper operator	Always use the proper operator for the data types involved. Always use the ampersand (&) when concatenating strings and the plus sign (+) when adding numeric values. If you use

	variants, it is possible to add two strings together and mistakenly produce a number instead of concatenating strings together.
Explicit conversion	Always explicitly convert values to the appropriate data types before operating on them. This is essential when working with variants. The **Val()** function is a good general purpose number conversion routine. However, the standard **CStr**, **CLng**, and **Ccur** functions are considered the most proper of all forms.
String comparisons	Using strings for comparisons is not recommended. If you must do this, however, I would suggest that you apply the **UCase(Trim$())** formula to both sides of the comparison. This will ensure a case-insensitive, nonwhite-spaced (leading or trailing) comparison.
Control usage	Never load list boxes, combo boxes, or grids with more than one hundred (100) static items. A good rule is that if there are more than fifty (50) items, use a virtual load or use a searchable system. One technique is to have a text box in which to enter the first few characters. Then when the focus leaves or the drop down is triggered, execute the search using the text as a filter and so limit the contents of the controls.
Control loading	Most of the time needed to load controls is required by the paint events as data changes. Because of this, it is wise when loading controls, such as list boxes or grids, to either make the control invisible or — better still — turn off the redrawing of the control. Then, once the data is properly loaded, the control can be redrawn. This can be done easily with the **SendMessage** API function.
Select Case	Using a **Select Case** instead of multiple **If** statements adds to readability and an increase in performance. **Select Case** statements are the preferred method for branching execution when there are more than three possible branches. Remember to cover all inputs. Always have an **Else Case** defined, and make sure that all criteria that can be handled are handled cleanly. Defined ranges are often helpful for covering all possible inputs.
Resource usage	When using file handles or memory, always close the files and release memory. When using file handles, call **FreeFile** to reserve a handle. Always restore system property settings if you use them.
Transaction usage	When using transactional logic or **BeginTrans** syntax, make sure that the logic begins and ends in the same function. For the **BeginTrans** syntax, this is required. For custom logic, it adds readability, consistency, and simplicity.
Loops	Every loop should have well-defined, obtainable, termination criteria. This is especially true in algorithms that require recursion.

Table 4. Variable Usage

Item	Comment
Array declaration	You should always use the lower to upper syntax as in **iArray(0 to 10)**. By explicitly defining the lower and upper bounds of static arrays, the impact of the **Option Base** statement can be removed.
Array traversal	Because of the flexibility in array dimensioning and the availability of the **Option Base** statement, always use the

	LBound() and **UBound()** functions as starting and ending points when traversing an array from top to bottom. It is *not* recommended that you store the upper and lower bounds in variables. The **LBound()** and **UBound()** functions are the only ways to guarantee accuracy through **ReDim** statements.
String literals	When using the same string literal more than three (3) times, you should implement the literal as a constant. This offers several benefits: compactness (constants require less memory for storage), efficiency (operations involving constants are much faster), and modularization (changes to the literal are confined to one place).
Scoping technique	A general rule of thumb when deciding on the scope of a variable is: Give it as little scope as possible. If the variable is used in only one function, limit the scope to that function. If it is only used in one module, limit the scope to that module. Consider reorganizing code to follow these techniques while ensuring that your organization fits the requirements of the layered paradigm.
Scoping initialization	Ensure that any variables that are not locally scoped are initialized before they are used. Variables that are of higher than local scope may be changed during the execution of a procedure.

Table 5. Syntax and Structure

Item	Comment
Variable declaration	Declare all variables on a separate line. Explicitly declare the type of each variable as you declare it. This enhances readability, offers easier debugging, and leads to fewer type definition errors.
Executable statements	All executable statements should be on separate lines. While the Visual Basic processor accepts several statements separated by colons included on the same line, it is considered bad form. You should adopt readability as an important goal for the structure and syntax of your code; therefore, don't use shortcuts like this to simply conserve space.
Goto statements	When writing structured code, avoid the use of **Goto** statements. There are special cases where the use of **Goto** statements can be considered acceptable and proper — for example, in error handling. However, you should limit your use of these statements to one or two labels per function and should uphold the guidelines on usability.
Parentheses	Use parentheses in statements with care. For example, you should always use parentheses to force the order of evaluation when necessary. You might also use parentheses when you decide that the code would be more readable because of tricky precedence or to qualify information for operations. In Visual Basic, the use of parentheses introduces an extra step in the operation. A memory copy is performed, and the remainder of the operation is applied to the copy of the result of the operation within the procedure. You can take advantage of this copy functionality when writing critical code to ensure that operations don't change the value of a variable, but operate on a copy of the variable instead. Of course, this advantage comes at a slight expense in the performance of operations.
Naming the main	When naming the main module of a project, always give the

module	primary module the same name as the executable name. This allows for cross-project consistency and for easier work transference. It establishes a common starting point for all applications.

Table 6. Formatting

Item	Comment
Consistency	Consistency is rule number one!
Headers	Every module and every function in every module should have a header. Some information for the header is: Description, Inputs, Outputs, External Function Calls, External References, Revision history. Comments are very important for understanding process flow and other settings, especially during debug, and maintenance phases of application development. Inline comments are good for covering key areas of complexity. Header comments are used primarily to outline complex algorithms or routines.
Error trapping	Every function should have an enabled error handler. In cases where failure is a simple exit, this lets you institute more complex error handling easily, with a minimum of changes.
Naming conventions	It is considered proper when naming functions and controls to use a naming convention. Some recommended conventions are <object><verb> or <verb><object>. The choice is usually dictated by the vertical organization of the relevant code. For example: If several routines to retrieve data for several objects are included in the same module, I suggest you use <verb><object> naming such as: **GetCustomer**, **GetProduct**, **GetOrders**, and so forth. However if the routines to support a single object all reside in the same module, it is considered proper to use <object><verb> naming such as: **CustomerLoad**, **CustomerUpdate**, **CustomerInsert**, and so forth.
Indentation	It's a good idea to indent your code to indicate the flow of control. Don't, however, indent the base level. The first level of code should be on the margin with no preceding tabs. Code that resides inside an **If...Then...Else** statement should be indented one tab.

Table 7. Procedure Organization

Item	Comment
Consistency	Consistency is rule number one!
Defined inputs and outputs for functions	Every function should have a defined range for each input and a defined range for acceptable success and error return values. It is critical that the return ranges be defined, attainable, and nonoverlapping.
Input value checking	Every procedure should perform range checking validation on all input values. This will often simplify algorithms and logic in the remainder of the routine. This practice will also allow you to catch and return error conditions as early as possible. By catching input range errors before commencing complex algorithms or logic, you can do cleanup and create graceful exit procedures with minimal logic impact.
Parameter lists	Procedures that perform similar functions or tasks should have similar parameter lists and output values. This level of consistency lends itself readily to reusable and maintainable

	code.
Modularize code	Code that is repeated or duplicated should be modularized. The general rule of thumb is that any code statement repeated more than three (3) times should be put in a separate routine. Similarly, code that exists in more than two (2) modules should be consolidated and generalized for reuse.
Procedure depth	Procedures calling other procedures should not require too many levels of procedure nesting. A general rule of thumb is that the procedure depth should not exceed six (6) levels.

Table 8. Writing Process

Item	Comment
Consistency	Consistency is rule number one!
Define coding goals	You should always have a defined list of objectives for each portion of code you write. Always reevaluate any code when you don't have a specific goal for its use. Never write code to provide a particular solution if the general case solution is as simple.
Generalization	Before writing new code, always clearly think through the uses for the code and consider whether you can write it in a reusable fashion. Custom code is usually the easiest to write, but is almost never the easiest to maintain or reuse.
Code reviews	Always review new code before releasing. Formal reviews are good, but not always practical. At a minimum, have all work read over by one of your peers to check for consistency and general appearance. Code that is poorly organized or isn't well commented can be spotted easily without intimate knowledge of the code. This quick once-over by another developer will serve as a good sanity check to prevent you from releasing substandard code to others. This is especially important in high stress situations with code that is quickly written. Never review more than two thousand (2000) lines of code at any one time: It's difficult to do a good job of reviewing such an unwieldy amount.
Code analysis	Code that is being put into production should be run through analyzers or profilers. If it is a front-end application for a database system, it should also be thoroughly benchmarked. Analyzers can tell a lot about the design of code by generating statistics about the code, such as executable-to-comment ratios, lines per module, local-to-global variable usage, literal usage frequency, and module level function break downs. Analyzers should also be used to generate call trees and relationship diagrams. These are helpful for graphically understanding the organization and structure of the code.
Code testing	Never test more than three (3) functions at a time. It is important for code coverage purposes that your testing be as focused as possible. By keeping the number of function jumps to a minimum, you can more effectively scrutinize the focus areas.

Class-level Concepts

These ideas are more suited to C++ developers, but with the changes in the Visual Basic language, developers using either language can benefit from these concepts.

Class Definitions

® Class interfaces should be as consistent as possible.

® Each class should describe a set of objects.

® Each class should serve a single, coherent, described purpose.

® Class interfaces should always be complete and as minimal as required.

® Classes should have as narrow a focus as possible.

® Base classes should contain only members that apply to all derivations.

® Base classes should contain only members that are used by all derivations.

® Classes should have as little coupling and interaction with other classes as possible.

Class Implementation

® Every class should have a copy constructor defined.

® Every class should have a defined assignment operator.

® Class constructors should put the constructed objects in well defined states.

® Class destructors should adequately clean up after themselves.

® Destructors in base classes should be defined as virtual.

® Destructors should never call anything that might raise errors unless they can be dealt with.

® Every class assignment operator should assign all members.

® Member functions should not return references to members less accessible than themselves.

Objects

® Objects should be designed for abstraction, not modularization.

® Collections of functions do not constitute an object.

® Objects that are independent should have independent behavior.

® Objects whose purpose is strictly management usually indicate poor design.

® Object states should be defined consistently.

® Object error states should always be strictly and completely defined.

® Special care should be taken when objects become multidimensional. These usually involve multiple versions or an element of temporal relationship.

® Objects that are services should not look for resources. Objects that are resources should register themselves with the services.

Relationships

® Take notice of the cardinality of defined relationships. They should always be delineated using the 1:1, 1:M, or M:M notations.

® The constraints on a relationship should always be strictly delineated along with the attributes of the constraint: Required, Unique, Distinct, Max, Min.

® Always distinguish an ISA relationship from an HASA relationship — to use Rumbaugh notation. (The notation isn't important, but distinguishing the relationship is.)

® Relationships should be defined inside Get and Set functions of the associated classes.

® Always consider both sides of a defined relationship. If one side is not implemented, this decision should be properly noted and explanations provided.

This technical article illustrates and discusses several fundamental concepts you should apply when developing client/server solutions using Microsoft Visual Basic. It is part of a series beginning with the article Client/Server Solutions: The Architecture Process, which is also available in the Microsoft Developer Network (MSDN) Library.

Overview

This article will discuss several basic concepts tailored to help you design high-quality Microsoft Visual Basic code. Although these concepts are useful for programming in many languages, I've adapted them to meet the specific limitations and use the strengths of Visual Basic. These basic concepts are:

® Consistency, consistency, consistency!

® Simple first, then elegant.

® Modular is better.

® Too much modularization is bad.

® Succinctness is a virtue.

® Recycle, recycle, recycle!

® Ask first, then code.

® Validate, validate, validate!

® Two is company, three is a crowd.

I've incorporated several examples throughout the following sections, but not every section has an example. For all sections, I suggest that you apply these ideas to existing code as a means of cross-examining the design that went into the code.

Consistency, Consistency, Consistency!

This should always be first and foremost in the thinking that goes into any design. As with most great concepts, this one is painfully obvious, but it is not practiced nearly enough. One reason for this is that different parts of code are often viewed from different perspectives by the same person. In the following example, a good naming convention is used for the variables, but the procedures have an obscure convention that isn't in line with the consistency of the variable names.

```
Dim iCustId As Integer
Dim sCustName As String
Global giProdId As Integer
Global gsProdName As String

Sub SaveCustomer()
Function RetrieveCustomerName()
Function UpdateAProduct()
Sub GetListOfProducts()
```

This same idea should apply to how we view controls, objects, stored procedures, tables, queries, and so on.

Another big advantage to consistency is the ability to use code generators much more effectively. Even code that performs different tasks using the same approach can easily be cloned; doing a simple search-and-replace is more advantageous than rewriting your code from scratch.

Simple First, Then Elegant

As the name implies, this concept is simple. You should write code first as simply as possible. When you do this, you can more easily see the limits and boundaries of the code and any errors present in the algorithm. Then later, as you polish the application and take advantage of performance improvements, you can modify individual functions with minimal impact, replacing simple routines with more elegant solutions.

There is a trap inherent in this concept that developers often get caught up in. Developers write simple functions with limited error handling, bounds checking, and so on, with the intention of returning at some later time to enhance the code and make it more robust. But as the project picks up speed, your "simple" code gets further and further from your mind. Finally, the project ships and you've never made time to go back through and clean

up your code.

This type of coding is not what is suggested here. The form of the code should be intact and error handling should always be a priority. But it is possible to use simpler algorithms — perhaps slower or more memory-intensive — to manipulate data. For example, you might first use a bubble sort because the code is compatible and easy to reuse. Later, when you have more time to work on performance, you can substitute a quick-sort routine. Or you might use some simple, slow paint algorithm and then later optimize it using the newest flashy paint algorithm if necessary.

Modular Is Better

The whole idea behind modularity is to organize code consistently. This concept supports the Layered Paradigm. No matter what architectural approach you use, your code should always be modularly organized. The main benefit is that your transactional logic will be clear and concise. Tasks that are similar or that are performed by several applications are easily navigated and maintained because they share the same organizational structure and transactional flow.

Too Much Modularization Is Bad

When organizing code, it is important not to overmodularize. Consider the following arrangement:

```
Sub FillFocusCustomerArray(ByVal iCustId As Integer)
...
    sCustName = GetCustomerName(iCustId)
    sCustPhone = GetCustomerPhone(iCustId)
...
End Sub

Function GetCustomerName(ByVal iCustId As Integer) As String
...
    For iIndex = LBound(sCust,1) To UBound(sCust,1)
        If Val(sCust(iIndex,0)) = iCustId Then
            GetCustomerName = sCust(iIndex, 1)
        End If
    Next iIndex ...
End Function

Function GetCustomerPhone(ByVal iCustId As Integer) As String
    ...
    For iIndex = LBound(sCust,1) To UBound(sCust,1)
        If Val(sCust(iIndex,0)) = iCustId Then
            GetCustomerPhone = sCust(iIndex, 6)
        End If
    Next iIndex ...
End Function
```

This is a simple example drawn from real project source code. The justification for writing the code this way was that when the name or phone number for a customer was needed elsewhere, it could be found simply by using the customer ID. However, because the code is so modularized, it is difficult to follow the flow of control. Excessive modularization also slows down the code and creates more code to be maintained. This follows the "modular is better" idea perfectly, but it doesn't enforce the "simple first, then elegant" concept. Consider a simpler implementation:

```
Sub FillCustFocus (ByVal iCustId As Integer)
.......
    For iIndex = LBound(sCust,1) To UBound(sCust,1)
        If Val(sCust(iIndex,0)) = iCustId Then
            sCustName = sCust(iIndex, 1)
            sCustPhone = sCust(iIndex, 6)
        End If
    Next iIndex
```

```
    ...
End Sub
```

This approach is simple and modularizes the task. However, it doesn't have the previous code's advantage of being able to retrieve an arbitrary name or phone number simply by using the customer ID. Once again, you can make a compromise to provide maximum flexibility with *quality* code. By changing the requirement a little, you can supply similar functionality. Consider the following approach:

```
Sub FillCustFocus (ByVal iCustId As Integer)
    ...
    For iIndex = LBound(sCust,1) To UBound(sCust,1)
        If Val(sCust(iIndex,0)) = iCustId Then
            sCustName = GetCustomerName(iIndex)
            sCustPhone = GetCustomerPhone(iIndex)
        End If
    Next iIndex
    ...
End Sub

Function GetCustomerName(ByVal iIndex As Integer) As String
    ...
    GetCustomerName = sCust(iIndex, 1)
    ...
End Function

Function GetCustomerPhone(ByVal iIndex As Integer) As String
    ...
    GetCustomerPhone = sCust(iIndex, 6)
    ...
End Function
```

This leaves a modular abstraction that gives us all the benefits of the "modular is better" idea, while maximizing the "simple first, then elegant" concept.

Succinctness Is a Virtue

Another important aspect of this concept is not to overdo the naming conventions. The general rule is to use common sense when naming objects and abide by the rule of consistency. Here are some other rules: Don't require children to encapsulate their parents (for example, column names shouldn't have table names in them always). Stored procedures don't need every sorting and grouping in the title. Controls don't need the form name in their name. Functions don't need to spell out every detail of what they do. Remember, be consistent.

Finally, don't create tasks that try to do too many different things in the same function. Break your tasks into pieces. This will help outline the transactional logic. It usually adds steps, but the steps can be made succinct enough to be worthwhile. Consider the following:

```
Function UpdCustomer() As Integer
...If Not GenMatchcodePart1() Then Exit Function
    If Not GenMatchcodePart2() Then Exit Function
    If Not GenTransaction1() Then Exit Function
    If Not GenTransaction2() Then Exit Function
    If Not SaveCustomerAddress() Then Exit Function
    If Not SaveCustomerTransaction() Then Exit Function
    If Not SaveCustomerData() Then Exit Function...
End Function
```

This code has been nicely modularized and looks clean, but this one function has a lot of conditions in it. If you ever need to change how a matchcode is generated (say now it only has one part in it), that would necessitate changing a critical transaction section. Consider the following:

```
Function DoUpdCust () As Integer
...If Not GenMC() Then Exit Function
    If Not GenTran () Then Exit Function
    If Not UpdCust () Then Exit Function...
End Function

Function GenMC() As Integer
...If Not GenMC1() Then Exit Function
    If Not GenMC2() Then Exit Function...
End Function

Function GenTran() As Integer
...If Not GenTran1() Then Exit Function
    If Not GenTran2() Then Exit Function...
End Function

Function UpdCust() As Integer
...If Not UpdCustAddress() Then Exit Function
    If Not UpdCustTran() Then Exit Function
    If Not UpdCustData() Then Exit Function...
End Function
```

In this implementation, it is a little easier to follow what is happening when stepping through the code. Also, at a later time, you'll have only one part to a matchcode or transaction generation that only touches one function. The same is true of the customer save facility. If you need to do some different processing (such as adding a **SaveCustomerDetail** function), you only have a single section outside the critical transaction loop.

Also, notice the shortened and standardized naming convention for functions: The **Do** prefix for transactional loops, the **Upd** prefix for a type of task.

Recycle, Recycle, Recycle!

Recycling is a major part of developing quality code. It consists of writing code that can be reused and reusing code that has already been written. The implementations of this code recycling are varied. Some developers might use libraries of old projects, or modules of code, or even just function snippets. Others choose to just cut and paste from work they have previously done. Still others borrow only from respected sources or from CDs of published code. There are many benefits of reuse. Because these benefits are listed everywhere (including earlier in this article), I won't go into it further.

Reusability is enhanced by writing modular code and by using consistent cross-application naming conventions. The implementation of this extends far beyond Visual Basic code into database code, stored procedures, triggers, and so on. After all, it is one thing to reuse pieces and parts — it is yet another to be able to reuse entire projects.

Ask First, Then Code

This is really about a consistent way to remember to recycle. Before new custom code is written, you should investigate existing sources of similar code or functionality. Even if the current code was written in a custom, nongeneric way, reusing it might save some typing. Also, the time savings might present an opportunity to modify the existing base into a reusable code fragment.

When writing code, remember to keep other uses for functions in mind and generalize accordingly. This is generally something everyone could do more of. The main roadblock to doing this is the "not written here" syndrome. Developers tend to be apprehensive about using code they didn't write. However, if everyone is using these concepts, then all code should be easy to use and maintain. Only by implementing these types of standards can developers begin to break down the roadblocks on the way to reusable, maintainable code.

Validate, Validate, Validate!

When writing modular code, it doesn't hurt to validate at every step of the way, from the front end, through the data layer, and into the back end. If the data validation rules have a low rate of change (ROC), then it is wise to build validation routines all the way through the code — even to the front end. If possible, load the data rules

from the back end or an .INI file.

When writing stored procedures, validate the data passed in before executing transactional logic. This will allow the procedure to better control error conditions. If all data has been pre-validated before an insert, no triggers will raise an error condition. This is not an argument for not using triggers. There should still be triggers — just don't count on them. By validating at every step in the process, you can maintain maximum control over the process flow at each step. This flow control lets you better pinpoint which pieces of data are causing problems at the earliest possible step in the process.

Using comprehensive validation techniques also opens new possibilities for constructing elegant error handling, logging, and reporting architectures. The flexibility and power of a stored procedure relative to a trigger allow for much more comprehensive error logging or returns.

Two Is Company, Three Is a Crowd

Write Boolean functions and procedures. This doesn't mean that all functions return true or false. But functions should always have a defined valid range and a defined invalid range (for example, pass or fail, true or false). This principle should extend to the database layer. Stored procedures and action queries should return values that fall inside or outside of a valid range. Consider the following code sample:

```
Function GetCustomer() As Integer
On Error Goto getcusterr
'Do some work

GetCustomer = iCustId
Exit Function

getcusterr:
    'handle error
    Error_Set
    GetCustomer = giErr 'some error number
    Exit Function

End Function
```

In this situation, you would expect to always get a valid customer ID or an error number. But without properly defining the interface, how would you be able to tell, inside the calling code, whether what was returned was an error number or a valid customer ID? And what does a zero value indicate? The trap here might not be obvious, but it can cause problems. Consider the following implementation:

```
Function GetCustomer() As Integer
GetCustomer = gicNoCust 'Some constant error number
On Error Goto getcusterr
'Do some work

If iCustId >= iCustIdMin AND iCustId <= iCustIdMax
    GetCustomer = iCustId
Else
    GetCustomer = gicInvCust 'Some constant error number
End If
Exit Function

getcusterr:
    'handle error
    Error_Set
    If giErr >= iCustIdMin AND giErr <= iCustIdMax
        GetCustomer = gicUknCust 'Some constant error number
    Else
        GetCustomer = giErr 'some error number
    End If
```

```
    Exit Function

End Function
```

This example is essentially the same. However, it is has been polished to include validation of the return data. This particular arrangement is a little bit of overkill, but it stresses the importance of validation and of distinction in returned values.

You can add code, but not a procedure, in the <INPUT> tag. A procedure must be placed in a <SCRIPT> section of a Web page, and referenced by the <INPUT> tag.

For more information, see Writing Event Procedures.

All client-side script is placed in a <SCRIPT> section of a Web page.

For more information, see Writing Event Procedures.

All script code is interpreted. Client-side script is run on the user's computer.

For more information, see Client vs. Server Scripting.

Script code is interpreted, not compiled.

For more information, see Client vs. Server Scripting.

Client-side script runs on the user's computer, and not on the Web server.

For more information, see Client vs. Server Scripting.

All script code is interpreted. Client-side script is run on the user's computer.

For more information, see Client vs. Server Scripting.

You can write event procedures for ActiveX controls.

For more information, see Writing Event Procedures.

Currently, Java applets do not trigger events.

For more information, see Writing Event Procedures.

You can write event procedures for hyperlinks.

For more information, see Writing Event Procedures.

You can write event procedures for standard HTML controls.

For more information, see Writing Event Procedures.

Controls on forms do not need to reference the frame.

For more information, see HTML Object Model.

If a control is on a form, you reference the name of the form and the name of the control.

For more information, see HTML Object Model.

If a control is on a form, you reference the name of the form and the name of the control.

For more information, see HTML Object Model.

If a control is on a form, you can refer to the control by referencing the name of the form and the name of the control.

For more information, see HTML Object Model.

The Microsoft Script Debugger debugs only client-side script.

For more information, see Debugging Script.

The Microsoft Script Debugger can be used to debug client-side script.

For more information, see Debugging Script.

The Project Explorer window of the Microsoft Script Debugger displays the relationship between frames.

For more information, see Debugging Script.

When a run-time error occurs on a Web page, the Microsoft Script Debugger launches and displays the location of the error.

For more information, see Debugging Script.

In JavaScript, all procedures are functions.

For more information, see JavaScript.

In JavaScript, all procedures are functions.

For more information, see JavaScript.

In JavaScript, you cannot use the naming convention *controlname_event* to create an event procedure that runs automatically when the event occurs.

Instead, you can set the event attribute of an object to call a function, as shown in the following script.

```
<INPUT TYPE=RADIO NAME=RadioGroup OnClick="ProcessOrder">
```

In JavaScript, all procedures are functions.

For more information, see Writing Event Procedures.

In JavaScript, all procedures are functions.

For more information, see JavaScript.

Browsers that support the <SCRIPT> tag will ignore comment tags.

For more information, see The <SCRIPT> Tag.

Browsers that support the <SCRIPT> tag will ignore comment tags and will include the script in the Web page.

For more information, see The <SCRIPT> Tag.

If a browser does not support the <SCRIPT> tag, it will display the script inside the <SCRIPT> section as text, unless the script is enclosed in comment tags.

For more information, see The <SCRIPT> Tag.

You enclose script in comment tags to prevent browsers that do not support the <SCRIPT> tag from displaying the script.

For more information, see The <SCRIPT> Tag.

The standard paradigm recommended in Visual Basic client/server applications is that of a series of layers. In this section, I will explain why I recommend the *Layered Paradigm* as a standard, what its strengths are, and where it breaks down.

The Layered Paradigm divides the *component operations* of an application into several component pieces. The component operation becomes the component level worker, as opposed to a *core function*. Before I talk about why this is best, here are some definitions for the terms I am using:

- *Core function*. For purposes of this article, this is the smallest unit of code. It encompasses object methods, subroutines, and functions. These functions are called by any element of the application that needs to use them. They are not restricted to a certain component area. This is the smallest unit that can be dealt with in a self-contained fashion (loaded, unloaded, modified, executed, and so on).
- *Component operation*. This is next lowest unit of code and includes any core functions called to complete a defined objective. *A component operation* directly supports a component area (**LoadProducts**, **SaveCustomer**, and so on).
- *Component area*. A component area can be composed of either component operations or core functions, or possibly both. It only denotes a specific division of labor. The main rule is that a component area must be modularly segmentable. In other words, it must be loaded and unloaded on deman; it must allow modifications to be made internally with no effect on the calling code; and it must have some concept of its own memory space.

  Another subdivision that will be used is that of *component module*. A component module is simply a component area that is organized so all supporting code exists in the same code .BAS file.

Developing an application built of components — object-oriented programming (OOP), Component Object Model (COM), and so forth — is a popular thing to do, but within the constraints of the Visual Basic programming arena, it can be exceedingly difficult. The performance and memory limitations of Visual Basic often stand in the way of developing successful applications based on component pieces. The Layered Paradigm works around this by providing a threshold that allows you to divide components into groups by their relative "atomicity." The Visual Basic programming arena requires a higher threshold when dividing component operations into a component's core functions, as opposed to the C or C++ programming arenas, where the construction of a component's core functions can be very atomic. By moving the threshold from an atomic core function to an atomic component operation, the application receives the benefits of component design while staying within the constraints of the Visual Basic programming arena.

Real-World Application

The smallest unit in Visual Basic that meets the above definition is the *module*. A module consists of code that is loaded and unloaded at the same time, can include the same scoping, and is modular in its interface. An OLE object instantiated from within Visual Basic code has similar characteristics. All the code in the object is loaded and unloaded with the instantiation of the object. It has its own scoping and has a modular interface.

Interfaces like these items, which are the lowest atomic units in the Visual Basic programming arena, can (and, under a true component model, would) be broken down into their functional pieces. For example, the **ExecSQLBool** function is unwrapped, so that the **SaveCustomer** method becomes a series of database transport function calls that together constitute the functionality of executing structured query language [SQL] statements to retrieve a Boolean result.

However, doing this isn't always an optimization. Often the application architecture will degrade in other areas (for example, every piece of the application that used **ExecSQLBool** now has to do its own database transport function calls). And what happens when the architecture changes—for example, if the database transport functions are now instantiated as methods on an OLE object? There would then be significant overhead involved with repeated instantiations of the SQL OLE object, compared to creating one instantiation and then passing everything through to that one instantiation.

When modifications are made to implement this new approach, it takes much longer because the database transport code is spread out through the application-specific code. When the modification is finally finished, there is only one instantiation. That object is passed around to all the components of the application, so they can use it directly. There is still substantial overhead involved with dereferencing the object with each pass to a different piece of the application, so performance is still not acceptable, and memory usage soars.

The Layered Paradigm, however, evens this scenario out nicely. Because there is only one instantiation, no other component has to spend time in dereferencing pointers, crossing application space, or other costly

maneuvers. All the code for a component operation (for example, the **ExecSQLBool** function, which is simply the making of a series of database transport function calls) is in one place, so any modifications are easy and localized. Memory is optimized because all the code for this component operation is unloaded when not being used. The Layered Paradigm provides all the benefits of a component design while side-stepping the unwanted frustrations that come from Visual Basic's limitations.

The Layered Paradigm has many benefits that help meet useful objectives in the changing world of client/server application development. Recognizable, measurable benefits include the following.

- *Maintainability*. Code modules in separate applications are organized in a recognizable manner. Component operation-oriented code is centrally located.
- *Reusability*. Component operation-oriented code is easily developed for reuse, specifically, component operations that cross application boundaries. The core functions that support a component operation are also good candidates for reuse.
- *Simplicity*. Modular design removes the use of "spaghetti code." It fosters elegance, not hacks.
- *Testability*. Modules can be tested easily. Modularization breaks up code tasks into smaller, more manageable units.
- *Size*. The memory footprint can be significantly reduced. Also, code size optimizations are much easier to implement and localize in a modular design.
- *Speed*. Modular code can be safely optimized without affecting the calling procedures.

As with any paradigm, there are also limitations. Some potential dangers are:

- *Functionalizing component operations*. This happens when component operations become so small that they are core functions in disguise. This leads to the same pitfalls as the functional component design we discussed previously.
- *Module overload*. This happens when the number of module component operations becomes too great to easily keep track of. This can be combated by grouping modular component operations that support the same component area into the same module. For example, rather than using 14 types of inserts, updates, and deletes, it would be preferable to move the component operations for a product into the same modular grouping. So the operations for an insert, update, and delete on a product will all go into a single product maintenance module. This would become a component module.
- *Mixing layer approaches*. This problem occurs when layers are mixed in some areas and not in others. For example, in the data interface for implementing product maintenance, the code might reference a window object. Conversely, in the data interface for implementing customer maintenance, the code does not. This inconsistency can be confusing and is not very maintainable. Consistency is the key to overcoming this, and a great many other programming trouble spots.

The standard layers of a Visual Basic application are:

® *User interface*. This layer is the topmost layer and is where all user interaction is completed. This layer supports no dependents and is directly dependent on the data interface. This layer is almost entirely nonreusable and is completely contained within the application boundary.

® *Data interface*. This layer is where all data is contained, or manipulated, in memory. This layer directly supports the user interface and is directly dependent on the transaction interface. This layer is mostly nonreusable and is usually contained completely within the application boundary. However in the case of distributed reusable component areas, the component might manage its own data.

® *Transaction interface*. This layer is where all transaction-based processing of data occurs. This layer coordinates all permanent storage of data, either through file or database access. This layer supports the data interface and is dependent on the external access interface. This layer may be reusable and may extend past the application boundary in the same way as the data interface.

® *External access interface*. This layer is responsible for all the communication of an application with external data sources. External data sources may be in the form of files, databases, or hardware. This layer supports the transaction interface and in most cases should be completely reusable. This layer might also be managed strictly by a component area that falls outside the component boundary.

A graphical representation of these layers follows (Figure 1).

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. An example of a layered architecture

The following coding example outlines how an update to a customer record is processed from the front end all the way through the database. This example is intended to show how the control of a process can be subdivided into the four (4) layers of the application.

® User Interface

```
Sub btUpdCustomer_Click ()
    'Begin the process of updating a customer
    btUpdCustomer.Enabled = False
    pnlStatus.Caption = Saving Customer...
    UpdateCustomer
    pnlStatus.Caption = "Ready"
    btUpdCustomer.Enabled = True
End Sub
```

® Data Interface

```
'This function verifies the data in the UI meets certain preliminary
' criteria for an update. Then it calls the update transaction.
Sub UpdateCustomer ()
Dim sMatchcode As String
Dim sCustName As String
...
    'Matchcodes must be = gicLenMC chars in length
    If Len(Trim$(txtMatchcode.Text)) <> gicLenMC Then
        'Call the error display module
        ErrDisplay "Matchcodes must be " & Str$(gicLenMC) & " chars in length."
        Exit Sub
    End If
    sMatchcode = Trim$(txtMatchcode.Text)
    ...
    if DoCustomerUpd (sMatchcode, sCustName, ...) then
    ...
End Sub
```

® Transaction Interface

```
'This function does the update of a customer.
```

```
Function DoCustomerUpd (ByVal sMatchcode As String, ...)
Dim sParam As String
DoCustomerUpd = False
'Build customer record as parameter list
sParam = "'" & Trim$( sMatchcode)
sParam = sParam & "' , '" & Trim$( sCustName)
...
'Validate the customer attributes
If ExecSQLBool(gscCustomerVal & sParam) Then
    'The customer record attributes are valid so...
    If ExecSQLBool(gscCustomerUpd & sParam) Then
        'The update was successful
        DoCustomerUpd = True
    Else
        'The update was NOT successful
        'Call the error display module
        ErrDisplay "The update to '" & sCustName & "' was NOT successful."
    End If
Else
    'The validation was NOT successful
    'Call the error display module
    ErrDisplay "The update to '" & sCustName & "' was NOT successful."
End If
...
```

® External Access Interface

```
Function ExecSQLBool (ByVal sQry As String) As Integer
    'Set the default
    ExecSQLBool = False
    'Check the connection
    If SQLDead(giSQLConn) = NUM_ERR_DEADCONN Then
        ...
    End If
    ...
    iRes = SQLCmd(giSQLConn, sQry)
    iRes = SQLExec(giSQLConn)
    Do Until (SQLResults(giSQLConn) = NOMORERESULTS)
    Do Until (SqlNextRow(giSQLConn) = NOMOREROWS)
            If Val(SQLData(giSQLConn, 1)) = NUM_EXEC_SUCCESS Then
            ExecSQLBool = True
            Else
            ExecSQLBool = False
                'Error data could be in column 2
            End If
        Loop
    Loop
End Function
```

This section includes the following topics:

® User Interface

® Data Interface

® Transaction Interface

® External Access Interface

® External Component Interfaces

The user interface is the only portion of the application that is responsive to user interaction. The user interface is where all data is presented to the user by means of window objects. The user interface is also where all inputs or modifications to data are made by means of window objects. The user interface should be one of two layers that references window objects (controls, forms, and so forth). In the best case scenario, the user interface would be the only layer that references window objects. However, in reality, allowing the data interface to reference the window objects directly can significantly reduce the amount of code and the simplicity of code paths. Since there are many aspects of a design that must all balance out (maintainability, reusability, performance, and simplicity), tradeoffs such as this are often considered.

What should the user interface include?

The user interface includes all event handlers or events. These subroutines are called in response to some user action (a click, a mouse move), to a change in status of a window object (**Form_Resize**, **Lost_Focus**), or to callback procedures (**VBSQL_Error**, **Timer**). It also includes procedures that either fill controls with data or retrieve data from controls. As stated above, the line between the user interface and the data interface is often very fine when discussing the issues of filling controls or retrieving data from controls. The responsibilities of the user interface include operations such as:

® Displaying application data or information via window objects.

® Responding to the changing states of window objects.

® Initiating user requests.

What should the user interface not include?

The user interface should not be responsible for operations such as:

® Interfacing with external data sources such as files, databases, and OLE objects.

® Performing lengthy data manipulations — for example, sorting arrays and writing registry entries.

® Enforcing business rules — for example, validating data and verifying logins.

® Enforcing business processes.

® Controlling flow of transactional logic.

The data interface is where an application completes all its in-memory data manipulation. The data interface is responsible for validating and manipulating all an application's data. The data interface supplies all data to the user interface for display in window objects. The data interface supplies all data to the transaction interface for use when supervising the external access interface. This would include SQL strings and parameters. The data interface may use *locally* stored data sources such as registry entries or caches to store operational parameters and data. However, all *external* access should be done through the transaction interface.

What should the data interface include?

The data interface includes any routines that will perform operations on the data of an application. The responsibilities of the data interface should include such operations as:

® Loading locally stored operational parameters (registry entries, .MDB file records).

® Sorting the data in arrays or structures.

® Validating the data in the structures.

® Overseeing the setting, or resetting, of operational parameters and work variables such as counters and flags.

® Formatting data for display in the user interface, including localization and masks.

® Formatting data for use in the transaction interface.

® Enforcing data interface business rules, such as the verification of data existence.

What should the data interface not include?

The data interface should not be responsible for such operations as:

® Displaying application data or information via window objects.

® Interfacing with external data sources such as files, databases, and OLE objects.

® Enforcing business processes.

® Controlling flow of transactional logic.

The transaction interface is part of the working internals of a client/server application. The transaction interface controls all data accessed by the application from an external data source. In addition, the transaction interface controls *all* updates to the data in the external data source that are initiated by the application. The transaction interface uses the external access interface to process its communication with the external data source and uses the data interface as its application data repository.

What should the transaction interface include?

The transaction interface oversees the external access interface in the transfer or manipulation of all data to and from an external data source. The responsibilities of the transaction interface should include such operations as:

® Initiating or building queries.

® Requesting information from the external access interface (**GetCustomer**, **InitLog**, and so on).

® Enforcing business processes.

® Handling violations of external access business rules, such as triggers firing or no write permissions.

® Controlling all transactional logic.

What should the transaction interface not include?

The transaction interface should not be responsible for such operations as:

® Displaying application data or information via window objects.

® Performing data manipulation operations such as sorting arrays or formatting data.

® Setting or resetting operational parameters and work variables such as counters and flags.

The external access interface embodies the communication of an application with an external data source. The external access interface is the specific transport or transports that the application uses to communicate with an external data source. Some common transports are DB-Library, Remote Data Objects (RDO), and Open Database Connectivity (ODBC). These transports require different code to complete specific functions. By encapsulating a series of external function calls or methods within modular functions, you can build reusable code that reliably completes certain component operations. For example, the component operation of logging into a database can be encapsulated in some general fashion. This same code can be used without modification in every application that uses an identical transport. The architecture used to call this function can be used even in applications that use a different transport. This preserves the consistency of a code base and allows for enhanced reusability and extensible architectures.

There are several benefits to coding a specific external access interface, the first of which is the demonstrated benefit of reusability. A second benefit derives from an application having a central pipe where all its external communication is completed. The ability to tap into a central location for logging and error handling can be a tremendous time saver.

A third major benefit is that encapsulating transport-specific code makes it extremely easy to replace an existing transport's code with code for a different transport. This method also allows an application to make modifications to the implementation of a transport without modifications to the application-specific code.

What should the external access interface include?

The external access interface directly handles *all* communication with an external data source. The responsibilities of the external access interface should include such operations as:

® Executing all queries.

® Retrieving all information from the external access interface such as **GetNextRecord** and **FilePositionSeek**.

® Passing along all external data source messages, errors, and so forth.

® Opening and closing files or databases — for example, **OpenFile** and **DBConnect**.

® Communicating with hardware, such as communication ports.

What should the external access interface not include?

The external access interface should not be responsible for such operations as:

® Displaying application data or information via window objects.

® Performing data manipulation operations such as sorting arrays or formatting data.

® Setting or resetting operational parameters and work variables such as counters and flags.

® Controlling flow of transactional logic.

External component interfaces are components that exist outside an application boundary and provide some service to those applications that instantiate them. External component interfaces are becoming much more common with the onset of OCX controls and OLE Server functionality in Visual Basic.

Where do external component interfaces fit into the Layered Paradigm?

With so much functionality being encapsulated and reused in these external component interfaces, the need to position these within the Layered Paradigm is very real. Fortunately, the Layered Paradigm facilitates the use of such components with remarkable ease. External component interfaces are positioned entirely based upon how much ownership they exert over their own display mechanisms, data storage and manipulation, transactional logic, and external data access. If an external component interface is responsible for its own data (for example, an object encapsulating customers), it might require an application's data interface to interface with its data. If it only manipulates data (for example, calculates tax for a purchase in a certain state), it might not require the data interface to interface with it at all. It might, however, require the transaction interface to interface with it (for example, generate a match code) or the external access interface to interface with it (for example, SQL OLE connection).

As should be obvious at this point, external component interfaces fit in smoothly with the Layered Paradigm. The only point that deserves special consideration is that of consistency. When you use an external component interface, I would recommend that your application implement a standard, reusable set of core functions if at all possible. This consistency throughout the application will add significant value when code bases are compared, code reused, and complex applications debugged.

As a C++ developer, you might find yourself initially at a loss when you begin to think about object-oriented design in Visual Basic. Since Visual Basic is not a true object-oriented language, how can you use it to implement a true object-oriented design? Furthermore, as Visual Basic becomes more object-oriented in future versions, how do you make sure that your implementation today doesn't preclude a safe and easy transition to the future versions of Visual Basic? The answer to these questions lies in the Layered Paradigm that I've presented here.

Of all the aspects of an object-oriented language, only data encapsulation has relevance in Visual Basic. (For example, Visual Basic classes have no ability to inherit functionality from one class to another. However, you can still provide an object-like interface by doing strong data encapsulation. Visual Basic has the ability to scope variables at a module level and allow private functions in a module. This enables you to define private data and methods that can help to encapsulate data from the rest of the application.

This approach fits well in a layered architecture, since the data interface of an application can be divided into separate components, each of which deals with a particular type of real-world data. In C++, objects encapsulate their data completely and often do not service user-interface commands — like filling a list box with a set of results. However, in Visual Basic, every time a function is called in a module, that module is loaded into memory, and when the function is finished, the module is unloaded. The overhead in calling a function is much greater than in C++, so tradeoffs must be made for performance reasons. The data interface can still strongly encapsulate its data, but it also should be able to accept list boxes and other user-interface controls to populate or to be able to pass entire structures to and from the user interface.

The Layered Paradigm uses three terms that may seem confusing to a C++ developer: *component*, *component operation*, and *core function*. These terms are intentionally different from common object-oriented terms because Visual Basic isn't as flexible as an object-oriented language, and design must be done in a slightly different way. However, they are fairly analogous to object-oriented concepts, and for the purposes of discussion, we will think of them in C++ terms as *object*, *object method*, and *nonspecific method*.

A component can be compared to a C++ object in the sense that it contains data (some private and some public) and methods to manipulate that data. However, it differs in that a component may be a single .BAS file and, in addition, has no ability to be instantiated as multiple objects, each of which contains its own copy of the object's data. Instead, a component contains just one copy of data or can contain an array of data structures, but it is entirely the responsibility of the component to maintain that data.

Component operations can be compared to C++ methods, but it is important to note that in Visual Basic they do not carry an implicit "this" pointer. Since that is the case, there is no reference to a specific instance of data. It, therefore, falls within the responsibility of the client code to let the component operation know which data to operate on. If a component is designed to support only one instantiation of its data, this construct is fairly simple, but if this component needs to have many instantiations of its data, then instantiations of data structures can serve to keep the data separate.

A core function can be compared to an unattached method in C++. This is a function that has no object reference and is independent of any particular data. It merely does a job and does not encapsulate any data. A C++ example of this would be a Windows API function. Though you could argue that all methods should belong to an object, there are times in Visual Basic where this approach is both unnecessary and improper.

In the end, at whatever level the Visual Basic application is designed, the architecture should always strive to encapsulate data and abstract complexity from the user interface. Using this approach will also decouple the user interface from the actual data format and will provide a scalable and maintainable design.

In the first article in this series, I discussed the *Layered Paradigm*, a standard paradigm I recommend for client/server applications built using Visual Basic. The Layered Paradigm is a method of application architecture that divides the component operations of an application into several component pieces.

Building an application using the Layered Paradigm can be complicated and challenging. Sometimes it is appropriate to design your application using vertical layering, and sometimes it is appropriate to use horizontal layering. In this article, I'll try to clarify how and when to choose each type of layering by illustrating a "best case" implementation approach.

The "best case" approach is not meant to be seen as the complete answer to all client/server problems. A "best case" solution is simply a solution that attempts to solve the major issues in a client/server scenario while continuing to provide leverage for creating more complex solutions. These solutions don't answer custom problems, and they don't implement a "pure object theory." They do, however, provide simple, elegant solutions to the most common problems faced by the client/server application developer. And they address these issues in a way that capitalizes on the benefits of the Layered Paradigm and good coding practices.

"Best case" solutions are solutions that address common real world problems and provide a customizable code base for creating custom solutions to real-world problems that are application-specific. In other words, "best case" solutions break all specific problems down into the parts they have in common with other problems. They then form solutions for the common parts only. By uniting the solutions for the common parts, they can more easily provide solutions to all parts of the specific problem and still provide a framework by which all specific problems that are similar may be solved.

"Best case" solutions handle the hardest problems as easily as a custom solution would while handling the common problems more easily than most approaches. "Best case" solutions also handle the easiest problems with tremendous improvements in all areas of the development and production cycle.

In summary, an application solution that uses "best case" methods will always be as good as a custom-coded approach given the same problem domains. And a solution based on common solutions will be an order of magnitude better for each facet of the common solution that is incorporated into the application-specific solution.

The goal of a "best case" solution is to solve the basic problem in general terms and then customize the common solution to solve the specific problem domain. The diagram in the next section illustrates some of the goals of software development and the areas that a "best case" solution addresses. It also shows which areas are addressed by custom implementation approaches.

"Best Case" Definitions

® *Design-time*. Time spent working out the details of the applications interfaces (Including the specifics of database access, transaction management, logging, error handling, etc.).

® *Maintainability*. The ease with which an application's code may be intelligently modified by a developer other than the original author. (Also, the ease with which it is possible to describe an application's architecture).

® *Performance*. The relative speed with which an application executes.

® *Portability*. The ability of an application to have internals modified without changes to the interfaces (for example, changing the database access method, logging method, processing algorithms, and so on).

® *Reuse*. The ease with which the code base can be used without modification in a similar situation — in other words, pure reuse in which the components are used in a new setting without modification. This term is used by contrast with *recycling*, the focus of "best case" solutions, which involves components being used in a new setting with only minor modifications to noncritical sections.

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. A graphical depiction of the effects of using a "Best Case" solution

As is shown by the illustration, any solution that will meet all five goals will cost more to develop for true reuse than the worth of the actual code developed (first-time usage only!). As any software engineer will agree, reusable code is always front-loaded, that is, it is substantially more costly to develop code for reuse than to develop code to solve a specific problem. However, notice that by using a "best case" solution, the design time was used in the best way and allowed the code to be more easily maintained and portable.

What Are the Implications of Implementing "Best Case" Solutions?

Essentially, the way to leverage your design-time forward through to portability and maintainability is through recycling. Recycling is designing and writing code to solve the common problems of client/server development without solving the specific problems of a particular application. This by no means excludes designing or writing code that solves specific problems. It just means that solutions are first created (or borrowed) to solve the most basic problems and then are slightly modified to meet the problem domain of a specific application.

Where Does "Best Case" Meet Reality?

A classic example is the loading of data from a database access call into a graphical list box control. This is done over and over in different problem domains, each time in a slightly different way. Some projects consolidate this into one call per form; others take the vertical approach and write a routine to load the control as a function that is part of some object grouping.

The first solution, using the Data Access Objects (DAO) as a transport for simplicity, writes a function like this:

```
Sub FillCustomerData()
```

```
...database stuff
Do Until ssTemp.EOF
    cmbStuff.AddItem ssTemp("StuffDescription")
    cmbStuff.ItemData(cmbStuff.NewIndex) = ssTemp("StuffId")
    ssTemp.MoveNext
Loop...more db stuff
Do Until ssTemp.EOF
    cmbJunk.AddItem ssTemp("JunkDescription")
    cmbJunk.ItemData(cmbJunk.NewIndex) = ssTemp("JunkId")
    ssTemp.MoveNext
Loop...
End Sub
```

This is encapsulation, right? The database access and the front end have been separated! Wrong. This is just organization. Encapsulation must provide some useful interface to distinct processes or entities within an application. This doesn't really do either, but it is a step in the right direction.

Our second solution writes functions like this:

```
Sub FillCustomerNameText()
Sub FillCustomerPhoneText()
Sub FillCustomerStuffCombo(ByVal iCustID As Integer)
...database stuff
Do Until ssTemp.EOF
    cmbStuff.AddItem GetCustomerName(iCustID)
    cmbStuff.ItemData(cmbStuff.NewIndex) = iCustID
    ssTemp.MoveNext
Loop
End Sub
Sub FillCustomerJunkCombo(ByVal iCustID As Integer)
...database stuff
Do Until ssTemp.EOF
    cmbJunk.AddItem GetCustomerJunk(iCustID)
    cmbJunk.ItemData(cmbJunk.NewIndex) = iCustID
    ssTemp.MoveNext
Loop
End Sub
```

Now surely this is encapsulation, right? The database access has been separated from the front end, and I now have a set of functions that encapsulates the customer object functions! Wrong. Once again in this example, we are just reorganizing code. We won't be able to consistently recycle the design or the code that went into this solution. We must still write the code for filling the combo boxes, and we must still completely walk through the design for an object with customer-like properties. We cannot reuse the implementation of the interfaces because the properties are slightly different. This solution also refuses to be recycled.

Isn't Pure Reuse the Best??

According to some reasoning, the encapsulation of the second solution above is more beneficial than any recycling. The argument some developers use is that because all the operations that operate on a customer have been encapsulated, pure reuse of this object will be possible. That argument has some merit. By tightly encapsulating a custom implementation, pure reuse does become possible. But with what benefit? Only an application that has exactly the same customer usage requirements can utilize this type of encapsulation. The many other applications that might benefit from leveraging some of the implementation of this design are simply unable to! This is because there are no recyclable qualities in the design. This type of implementation only makes it harder for any reuse (especially the impure type!) to occur from application to application or even within an application.

Recycling doesn't strive to support pure reuse, but rather to provide a code base that can be leveraged for solutions to similar but different problem domains. Consider the following solution:

```
Sub FillCustomerData()
...some code to get customer ids and keys
txtCustName = ExecSQLGetText(sGetCustomerName & sKey)
ExecSQLFillLBItem cmbStuff, sGetCustomerStuff & sStuffKey
ExecSQLFillLBItem cmbJunk, sGetCustomerStuff & sJunkKey
End Sub
```

This would be a nonoptimized solution without object encapsulation. Notice that there would have been defined a standard for database access that exposed the functions called here. Since this solution obviously does not show all object operations being encapsulated, here is the revised solution using object encapsulation functions:

```
Sub FillCustomerData(ByVal iCustId As Integer)
CustomerNameGet txtCustName, iCustId
CustomerStuffGet cmbStuff, iCustId
CustomerJunkGet cmbJunk, iCustId
End Sub

Sub CustomerNameGet(ctlMe as Control, ByVal iCustId As Integer)
Dim sQry as String
sQry = GetNameQry(iCustId)
ExecSQLFillCtl ctlMe, sQry & Str$(iCustId)
End Sub

Sub CustomerStuffGet( ctlMe as Control, ByVal iCustId as Integer)
Dim sKey as String
Dim sQry as String
sKey = GetStuffKey(iCustId)
sQry = GetStuffQry(iCustId)
ExecSQLFillLBItem ctlMe, sQry & sKey
End Sub
```

In this example, the standard database calls would be encapsulated in customer object operation functions. Using this technique, the only knowledge about customer object structure would reside in these customer object functions.

These examples do not specifically make use of known "best case" solutions; however, they do emphasize that by using a recyclable implementation, it is possible to leverage any code base and still maintain the benefits of encapsulation and object orientation. Recycling is the essence of a "best case" solution. All "best case" solutions are recyclable by definition.

Recycling doesn't just impact how functions are organized and where data is stored. It impacts how data is stored, the naming conventions used for all parts of the system (database objects, stored procedures, modules, forms, and controls), the functionality of stored procedures and queries that are executed, and the organization of the business and transactional logic of the application.

A Recycling Example

Consider a simple database front end. This front end accesses a SQL Server by executing stored procedures using DAO as a transport. The focus of the front end is to modify the records for several "objects" in the database. For the sake of simplicity, we will define the objects as follows:

® *Resource*. Each instance has a name, an associated skill, an associated manager, and an associated task.

® *Skill*. Each instance has a name.

® *Manager*. Each instance has a name and a title.

® *Task*. Each instance has a name, a start date, and a duration.

Simply put, the front end must have the following features (in no particular order, but numbered so that the discussion that follows can refer back to them):

1. A multiple-document interface (MDI).
2. The ability to perform login procedures to any designated SQL Server.
3. The ability to check front end version with database version upon database connect.
4. The ability to log all errors to a file.
5. The ability to log all SQL statements to a file.
6. A child window to view, insert, edit, and delete resources.
7. A child window to view, insert, edit, and delete skills.
8. A child window to view, insert, edit, and delete managers.
9. A child window to view, insert, edit, and delete tasks.

This situation is about as common as can be found. I have deliberately defined it in simple terms in order to emphasize the thought process involved in breaking the components down. In the next section, I will discuss how to engineer a "best case" solution for this application.

The Recycling Solution: Step One

The first step in engineering a "best case" solution is to identify any common problems in the specific application problem domain. In this example there are several. In fact, every single requirement is in some way a common problem! Requirements 1 through 5 are common to client/server applications. Requirements 6 through 9 are common to each other. The functionality of requirements 6 through 9 is also common to client/server applications. Being able to see these commonalties is the first step towards engineering a recyclable solution.

The Recycling Solution: Leap Two

The second phase is to design the components that will provide solutions to the common problems. Based on the previous requirements list, I will condense these into the common problems identified as follows (in no particular order, but numbered so that the discussion that follows can refer back to them):

1. Manage an MDI interface, that is, status bars, toolbars, menus, and so forth.
2. Provide a generic login window to collect login information for server systems as well as files.
3. Store and retrieve version information from a generic database object (table).
4. Manage a log file for errors.
5. Manage a log file for SQL statements.
6. Manage child windows.
7. Retrieve data from database.
8. Manage database transactions.

In the following paragraphs, I will outline some methods for addressing these issues. It's important to note that the key component in this phase of the recycling process is to take the current set of requirements and condense them. (Keep in mind that these concepts do not comprise the only solution — just one of many solutions.)

1. Provide a generic MDI parent

To do this may require addressing the following issues:

® Define MDI interface variables.

® Define configuration operations.

® Define generic status bar configuration and operations.

® Define generic toolbar configuration and operations.

® Define generic menu subset.

Providing a generic MDI parent is done by specifying variables and functions or subroutines by which the MDI parent can configure itself at run time. For example, if the behavior of a toolbar is to be configurable, a routine to set the properties of the toolbar should be defined. If there are standard variable properties that may be used (like constants for the status display or a flag for an application wide setting), these should be defined. Instead of just hard-coding values in place or writing a custom toolbar configuration routine, think about how it can be generalized — that is, identify the tasks that will have to be done from application to application. Then, consolidate these types of operations.

2. Provide a generic login window

To do this may require addressing the following issues:

® Define login window variables.

® Define configuration operations.

® Define interface for application interaction.

Providing a generic login window is done by specifying what the standard login window will look like and how it can be configured. For example, what controls are displayed for a server type of login as opposed to a file type login. Are there graphics or information (like version numbers) that need to be displayed? These interfaces should all be defined in a generic sense. Don't hard-code versions or captions! Provide a function or variables to set the properties of the window. Expose an interface to decide at run time which controls to display. If it can be generalized once, it will not need to be done again.

3. Define a version control system

To do this may require addressing the following issues:

® Define what information to store with the version (build numbers, debug flags, and so on).

® Define how the version is stored (numbers, a string, in a table, or a stored procedure).

® Define the mechanism to communicate the version information to the front end (stored procedure or SQL statement to select from a table).

It is important to ensure that a solution involving an external system (SQL Server or Microsoft Access file) be abstracted enough to allow a system substitution with little impact on the application. If it is conceivable that the application may need to utilize both a Microsoft Access file and a SQL Server database, then storing any version information in a stored procedure would be portable from the SQL Server implementation only in the form of a predefined Microsoft Access query. This may not have the features available in a stored procedure; therefore, this type of generalization is not recommended. Recycling solutions should always be designed in as abstract a way as possible. Recycling solutions always keep in mind component substitutions. Being able to keep these substitutions in mind is why properly defining interfaces is so essential to creating a recycling solution.

4 & 5. Manage log files

To do this may require addressing the following issues:

® Define a mechanism to manage file access and existence.

® Define a mechanism to write a data stream to the file.

® Define appropriate interfaces to these mechanisms.

As should be apparent, the commonality between problems 4 and 5 was identified. Because the task is the same and only the data content is different, one and only one solution is required. This is a very good expression of recycling at work. By defining one solution with enough abstraction, multiple problems can be solved simply and elegantly. The added benefits are that the solution is consistent through all uses. The interface to an error log is the same as the interface to the SQL statement log. We have a simple solution for two problems, and we get maintainability and portability for free! This type of solution consolidation will be discussed again later.

6. Manage child windows

To do this may require addressing the following issue:

® Define a mechanism to instantiate and destroy child windows.

® Define a mechanism to configure and control child windows.

Managing child windows is simply thinking through a standard, recyclable way in which a configurable number of child windows may be managed and instantiated. This section might actually be designed as a function of the MDI parent window. There are many schemas for this — the idea is to choose the one that solves the common problems and then to use that solution to leverage a custom solution.

7. Retrieve data from database

To do this may require addressing the following issues:

® Identify the standard data retrieval tasks.

® Identify the functionality of the standard data retrieval tasks.

® Identify the interfaces to the standard data retrieval tasks.

® Identify any components that may be utilized to provide these tasks.

Retrieving data from a database is the second biggest problem that is addressed in client/server development. Because it is such a large section of the development work, it deserves the most consideration. For now, the following summary will suffice to present the general concept.

In a well designed client/server system, there are several main categories of data access models that are common in client/server applications. They are as follows:

® Online data manipulation

® Online querying

® Online querying with local cache

This particular application is a simple online data manipulation tool, and the discussion will focus on that specific category. Within that category, there are several basic types of data retrieval that are commonly done in a client/server system. They are identified for this discussion as follows:

® Retrieving a single piece of data

® Retrieving a single row of multiple columns

® Retrieving a set of multiple columns and rows

With the common data retrieval methods identified, the next step towards implementation is to classify the variations that will be required for the solution to become recyclable. To accomplish this, the common tasks that these types of data retrieval support must be identified. By using the term *common*, it is implied that these are tasks that may be repeated and may be implemented in a configurable way. Customized implementation is not the goal of this step. The following is a simple breakdown of the required tasks:

® Retrieve a single piece of data into a string.

® Retrieve a single piece of data into a control.

® Retrieve a single row of multiple columns into an array.

® Retrieve a set of multiple columns and rows into an array.

® Retrieve a set of multiple columns and rows into a list control.

The previous step is included simply to show the progression from concept to design and then to implementation. It was *not* meant to be the complete working through of a data retrieval architecture. For this level of detail, see the section External Access Interface in the previous article in this series, "Client/Server Solutions: The Architecture Process."

8. Manage database transactions

To do this may require addressing the following issues:

® Identify the standard database transaction tasks.

® Identify the functionality of the standard database transaction tasks.

® Identify the interfaces to the standard database transaction tasks.

This problem is the single biggest problem addressed in client/server development. Because it involves such a large portion of the development effort, we won't in this discussion attempt to address all the possible ramifications of this problem. This discussion and the following section will, however, illustrate how to engineer a solution for this particular application.

Since the external database component (SQL Server) of this particular application supports stored procedures, the interface to database transactions can be drastically simplified. The following is a list of each major type of transaction and its interface specification:

® *Insert*. Short form "Ins". Takes the data values. Returns a zero success code; all non-zero returns are considered error conditions.

® *Update*. Short form "Upd". Takes the key and the data values. Returns a zero success code; all non-zero returns are considered error conditions.

® *Delete*. Short form "Del". Takes the integer key. Returns a zero success code; all non-zero returns are considered error conditions.

So the actual designed interface at the database access level would simply be one routine: Execute the procedure and return the code. If the return is zero, indicate success; if the return is non-zero, handle as an error condition. (This is the hook for adding localization or error lookups.)

Seeing the impact of decisions like this one allows for effective recycling on a project-to-project basis. The crucial leap is to see that the first and most important step is always simplification down to a base form. Customization at any time afterwards becomes merely an academic exercise. The standards for database access, for form-level organization, for service management, and all other components, are set forth by simplifying common problems. Once the standards are in place, it is possible to capitalize on them in a recyclable way — though not necessarily in a reusable way. Of course, solutions that are simplified enough and common enough can be made reusable. The path to making these solutions reusable becomes that much shorter because the exercise of generalization (which is the most complex part of the process) has already been accomplished. But unlike reuse, the solution is not committed to 100 percent reuse. At no time does the situation become an all-or-nothing decision. The solution can be recycled as much or as little as desirable with absolutely no impact on other users or implementations of the solution.

The Recycling Example Concluded

This example is not the only possible solution to the proposed requirements. It simply illustrates the processes, concepts, and priorities that, taken as a whole, comprise the recycling approach. Following are some guidelines to consider.

® Set standards for your development team, your project, and any group associations. Choosing a standard set of procedure types, transaction groups, function organizations, forms design, and so forth will help your development effort in several ways:

  — It will foster the recycling of solutions because the code will require only minor modifications.

  — It will subtly influence individuals to code according to the standards by their use of recycled solutions.

  — It will increase the communication potential within your team, project, and groups because common techniques and organizations are done across the board.

- It will increase common "look and feel" in all aspects of your development effort. Each piece — forms design, database object scripts and layout, or code organization — will have its place in the standard.

® Carry standards through all areas of the development effort.

By setting standards for the database perspective that work for the front-end coders, it provides a much more efficient means of communication between development areas. Front-end coders will already be familiar with the transaction mechanism used and will not require re-education for each project. The same is true for other areas of the development effort.

By using standards in one area that match standards in another area, efficiency and communication is maximized. Use simple approaches like defining a "list" query. It returns data in a certain format. It starts with a specific prefix. The back-end developers can anticipate its creation and the front-end developers can provide a generic routine to invoke the query and return results to form controls.

This communication of standards invokes a synergy between development areas that can smooth over some of the rougher issues involved in making areas with different tasks focus on the same goal.

® Develop aids to help in the creation of recyclable code.

Setting standards means you can use CASE tools and code generators to generate code and templates.

This will impact not just front-end code (Visual Basic, C++), but will hold true for SQL scripts, build and make files, and documentation as well.

® Devise a mechanism for evangelizing recycled solutions.

Having an individual or group tasked with evangelizing recycled code provides insight into the development effort that has lots of possibilities. This group can be responsible for formally reviewing code and documentation. This group can be responsible for evaluating new "off the shelf" solutions like CASE tools, methodologies, code generators, and templates.

In the end, this group can transform into the group that manages reusable code bases as well. This will provide an easy inroad for the group to establish contacts, open dialogue, and set up work practices.

A Visual InterDev Web project is not a Web site. It keeps track of the files that make up a Web site.

For more information, see Creating a New Project.

A Visual InterDev Web project contains all of the files used to create a Web site.

For more information, see Creating a New Project.

A Visual InterDev Web project contains all of the files used to create a Web site, not just the .gif and .jpg files.

For more information, see Creating a New Project.

A Visual InterDev Web project contains all of the files used to create a Web site, not just the .asp files.

For more information, see Creating a New Project.

Active Server Pages can contain both server-side and client-side script.

For more information, see Authoring Active Server Pages.

Active Server Pages can contain HTML <SCRIPT> sections with client-side script. They can also contain <SCRIPT> sections with the RUNAT attribute set to Server that contain server-side script.

For more information, see Authoring Active Server Pages.

Active Server Pages are processed by the Web server. The server-side script in Active Server Pages is run before a response is returned to the client. HTML pages do not need to be processed by the Web server before they are returned to the client.

For more information, see Authoring Active Server Pages.

The FrontPage Editor cannot be used to edit Active Server Pages.

For more information, see Authoring Active Server Pages.

If you plan to install components on computers other than the Web server, and call them remotely, you will need to create .vbr files to register the components remotely. For more information on distributing components over multiple computers, see Installing Remote Automation and Distributed COM Components in Visual Basic Books Online.

The syntax <%= i+i %> displays the value of the expression i+i, or 10.

For more information, see Active Server Page Syntax.

The syntax <%= i %> displays the value of the variable i, or 5. The syntax <%= i+i %> displays the value of the expression i+i, or 10.

For more information, see Active Server Page Syntax.

The syntax <%= i %> displays the value of the variable i, or 5. The syntax <%= i+i %> displays the value of the expression i+i, or 10.

For more information, see Active Server Page Syntax.

The syntax <%= i %> displays the value of the variable i, or 5.

For more information, see Active Server Page Syntax.

A <FRAMESET> tag placed inside a <FRAMESET> tag creates a nested frame.

For more information, see Nesting Frames.

A <FRAME> tag defines the source file for a frame in a <FRAMESET> tag.

For more information, see Nesting Frames.

The <IFRAME> tag defines a floating frame outside of a <FRAMESET> tag.

For more information, see Nesting Frames.

A <FRAMESET> tag placed inside a <FRAMESET> tag creates a nested frame.

For more information, see Nesting Frames.

License Package (.lpk) files contain license information about the ActiveX controls that are used in a Web page.

For more information, see Creating HTML Forms.

Forms are not visible to Web browsers.

For more information, see Creating HTML Forms.

HTML Layout (.alx) files contain layout information about the placement of controls. An .alx file can be used in multiple source files.

For more information, see Creating HTML Forms.

When a user clicks the **Submit** button, forms automatically send the data in controls to the Web server.

For more information, see Creating HTML Forms.

You create a multi-select list box with the <SELECT MULTIPLE ...> tag.

For more information, see Adding Standard HTML Controls.

The standard HTML controls are text box, two types of list boxes, check box, radio button, hidden control, and three types of buttons.

For more information, see Adding Standard HTML Controls.

You create a check box by using the HTML <INPUT TYPE=CHECKBOX ...> tag.

For more information, see Adding Standard HTML Controls.

You create a command button with the HTML <INPUT TYPE=BUTTON ...> tag.

For more information, see Adding Standard HTML Controls.

You can edit .gif files with an image editor such as Microsoft Image Composer.

For more information, see Authoring Static HTML Pages.

You can edit .asp files with the Visual InterDev Source Editor.

For more information, see Authoring Static HTML Pages.

You can create .ocx files by using Microsoft Visual C++.

For more information, see Authoring Static HTML Pages.

You can edit HTML files by using the FrontPage Editor.

For more information, see Authoring Static HTML Pages.

Article ID: Q143090
Creation Date: 28-JAN-1996
Revision Date: 16-DEC-1996

The information in this article applies to:

® Microsoft FrontPage versions 1.0, 1.1

® Microsoft FrontPage versions 1.0, 1.1, 97

Summary

This article describes the WebBot component of FrontPage and it lists the WebBot components that are available in FrontPage. A WebBot component is a dynamic object on a Web page that is evaluated when the page is saved or, in some cases, when the page is viewed in a browser. The following WebBot components are available in FrontPage:

- Annotation
- Confirmation Field
- Discussion
- HTML Markup
- Include
- Registration
- Save Results
- Scheduled Image
- Scheduled Include
- Search
- Substitution
- Table of Contents
- Timestamp

More Information

WebBot Annotation

The WebBot Annotation allows you to insert annotations or comments on your Web page. These annotations can be viewed from the FrontPage Editor; they do not appear in Web browsers. That is, when a page containing annotation text is displayed in a Web browser, the annotation text is invisible and does not take up space on the page. Annotation text is displayed in purple in the FrontPage Editor, and retains the character size and other attributes of the current paragraph style.

WebBot Confirmation Field

The WebBot Confirmation Field is replaced with the contents of a form field. It can be used to echo information, such as a user's name or e-mail address, back to the user as confirmation of the values they enter in a form.

Some uses for the WebBot Confirmation Field are: echoing the user's name and password for a restricted Web, and verifying the name and mailing address for a user who has submitted a form.

WebBot Discussion

A WebBot Discussion allows you to set up a discussion Web page. The WebBot Discussion collects information from a form, formats it into an HTML page, and then adds the page to a table of contents and to a text index. The WebBot Discussion gathers information from the form and stores it in one of a selection of formats, such as HTML or ASCII text. The WebBot Discussion is accessed from the Form Properties dialog box.

WebBot HTML Markup

The WebBot HTML Markup allows you to enter HTML tags that are not directly supported or available in the

FrontPage Editor. For example, the FrontPage Editor does not include support for table elements or the marquee element. To enter either of these elements in the FrontPage Editor, you must use the WebBot HTML Markup.

Note Because the FrontPage Editor does not support these elements, it will not import them. These elements will be preserved on export only. FrontPage does not verify that your markup is valid HTML.

WebBot Include

The WebBot Include is designed to replace itself with the contents of another file. The most common use for the WebBot Include is to place the same text or graphic on every page in a multi-page Web. Examples of items that you might want to display on each page in your Web include: copyright information, headers and footers, and product logos. The WebBot Include requires the URL of the file that contains the text and images to be included on a page. Any modifications to the source document are automatically sent to each page for which it is included.

WebBot Registration

The WebBot Registration allows you to automatically register users for access to a service. The WebBot Registration adds the user to the services's authentication database; it can then gather information from the form and store it in one of a selection of formats (HTML or ASCII, for example). The WebBot Registration is accessed from the Form Properties dialog box.

WebBot Save Results

The WebBot Save Results gathers information from a form and stores it in one of a selection of formats (HTML, ASCII text). When you submit the form, the WebBot Save Results appends the form information to a specified file in a specified format. The WebBot Save Results is accessed from the Form Properties dialog box.

WebBot Scheduled Image

The WebBot Scheduled Image is used when you plan to replace one image with another within a specific time period. On the expiration date, the original image is no longer displayed and another image takes its place or the place where the image existed is left blank.

When you insert a WebBot Scheduled Image, you will need to enter the name of the image and any alternate image to be included on the page. The image must be from the current Web.

A WebBot Scheduled Image runs when a change is made to your Web. To ensure proper timing of the WebBot Scheduled Image, make some changes to your Web daily.

Scheduled WebBot Include

The Scheduled WebBot Include is used to insert the contents of a file during a specified period of time. When the time period expires, the contents of the file are no longer displayed. You can specify an optional page to be included after a given date.

A Scheduled WebBot Include is run when a change is made to your Web. To ensure proper timing of the Scheduled WebBot Include, make some changes to your Web daily.

WebBot Search

Use the WebBot Search to create a search form. The FrontPage Server extensions maintain a list of words found in each page in a Web. If the Web includes a discussion group, FrontPage also maintains a dynamic list of words found in each entry of the discussion group. At runtime, the search form created by the WebBot Search returns a list of pages containing the words the user entered in the form. The words can include Boolean keywords, such as and, not, or, and parentheses.

WebBot Substitution

The WebBot Substitution allows you to type an abbreviation that is substituted by its expanded form. For example, you can set a variable, such as HTML, to represent "Hypertext Markup Language" (without the quotation marks). Then, instead of writing out "Hypertext Markup Language," you could type HTML, and HTML

will expand to "Hypertext Markup Language" at runtime. You define the variable as a parameter in the Web Settings dialog box (on the Tools menu, click Web Settings) in the FrontPage Explorer.

WebBot Table of Contents

The WebBot Table of Contents generates an outline of your Web. The WebBot Table of Contents updates this outline each time the Web content changes. It does not create a table of contents based on bookmarks. It uses the title element of the subpages to generate the table of contents.

WebBot Timestamp

The WebBot Timestamp inserts a date and time stamp that corresponds to the last time the page was edited or automatically updated.

KBCategory: kbusage
KBSubcategory:

References

Additional reference words:

- 1.00 front page Vermeer
- substitution
- substituting
- replacing
- confirmation
- confirming
- reply
- echo
- annotate
- annotation comment

You can view client-side scripting and server-side scripting as two very different approaches to Internet application development even though they share many similarities. In order to compare these approaches, I built a pair of applications that represent the opposing techniques while providing basically the same user function. Both of the applications process simple banking transactions over the Internet using data services provided by a Component Object Model (COM) server, and they both use VBScript (Visual Basic Scripting Edition) code within Internet documents as the application glue. The ActiveX scripting engine in Microsoft Internet Explorer version 3.0 evaluates the script code contained in the client-side version while the ActiveX scripting engine in Microsoft Internet Information Server (IIS) evaluates the scripts contained in the server-side version. The sample files include the data services provider as well as all of the Internet files for both applications. While I developed the sample using Microsoft Windows NT version 4.0 server and workstation, other client platforms are compatible depending on the scripting technique used.

This white paper includes the following topics:

A control palette is an ActiveX control that wraps up the functionality of both the user-interface and data manipulation routines. Now that Microsoft Visual Basic version 5.0 allows you to create your own controls, you can take a user-interface and turn it into your own drag-and-drop control — a window into your data at the click of a button. This article examines the design and development process that goes into creating a control palette. Additionally, this article guides you through the creation of a control palette called BankView, which allows you to display and navigate bank and account information. And most importantly, the article shows you how to create your control using a robust, layered, client/server methodology.

This white paper includes the following topics:

With the advent of ActiveX controls, and the onslaught of tools that make control creation easier, the task of creating user interfaces continues to shrink in complexity. In other articles, you've seen how to create and script controls. You may have seen how to design controls for easy programmability, and even how to take advantage of automatic download functionality so users can get and dispense your controls easily. In this article, I show you how to take this ease of use a little farther.

A Word About Palettes

Instead of simply creating controls that wrap some bit of algorithmic functionality, and expose that functionality through automation, I'll show you how to create controls that interact with other controls in an automated way. This isn't the same as creating controls that perform client/server tasks, like the controls that were developed in the article, Designing Intelligent Control Palettes. In that article, the concept of a control is extended from the notion of publishing algorithmic functionality to publishing data from a client/server system. A control palette isn't just a generic control for data access, like in a data window, or data-bound control. A control palette actually provides a custom interface to a unique system. The strength of a control palette is that it isn't simply a more specialized version of a generic control, like a special form of listview, or a data-bound tree control. A control palette isn't programmed, it is given minimal information, and then simply exists. An example is the Bank editor control palette first put together in "Designing Intelligent Control Palettes." However, while this level of specialty is useful in many situations, in the case of enterprise systems there are other design approaches that can be equally rewarding. This article approaches one of those and shows you the design process that went into it.

Interconnecting Palettes

Creating a good user interface is hard. Enforcing user interface standards across multiple applications is even harder. Reusing user interface components is almost impossible. Even today with ActiveX controls, developers often only focus on creating special versions of standard controls to add or enhance the functionality in their applications. However, creating interconnecting palettes is more than that.

Think about the number of systems in your average enterprise and how many of those systems use common or related elements — like address information. How many systems and front ends does your company have (or have you written) to display address information? Usually there is that one good one and others of varying quality, one for every application you use. Okay, so it's obvious that a company can have lots of different user interfaces. Now what about the storage format and manipulation routines for those interfaces. How vastly different and unique are they? If your company is normal, I'm sure you have one for each interface as well. Here is where a control palette would be useful. You could take the best bits of the best implementations and put together the best address control around, using standard storage and manipulation routines. It would be absolutely incredible! Until you try and integrate this control with all the different systems, each with their own implementation of a person, or customer, or business. All that user interface code to touch! What a pain. Of course, if your person, or customer, or business was also a control, you could very easily update it to work with a new address control instead of driving a bunch of separate text box and combo box controls. Think of all that weird, non-standard user-interface code you could get rid of. See where this is going?

In today's enterprise, having only one way to get at your data is unrealistic. You need components that allow you to build a custom interface without having to really build a custom interface. Specialized versions of tree controls are useful, but standard access to customer data, both in the back end and the front end, is the really powerful side of palettes. And as you see from above, just creating palettes for your user-interface components isn't enough. To be really snap-together compatible and truly reusable, the palettes you build have to have the ability to take advantage of other palettes. Truly reusable user interfaces don't need code or even scripting to drive them. They simply are connected in the same space with other controls and they work together seamlessly.

Think about the last time you used a third-party control. You had to figure out a programming model to drive the control. You had to know what methods to call, what properties to set, and what events were available. With interconnected controls, you don't need that. You simply connect them together, point them at their data, and they drive each other and themselves. At least that's the idea.

Next, you can see a working example of some interconnected controls. So as not to introduce a new model, this article will start with the Bank control palette from "Designing Intelligent Control Palettes," and add and remove some stuff to create a working bank system. Here's the control in its editor state:

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. The BankEx control.

This version is called BankEx for Bank with extended functionality. I left it intact so you would know I've really

started with a fully-functional control palette. I'll hack it down to just the essentials for the rest of the sample. In your designs there will be a time when you need the full-featured, self-contained palette. There are other times when you want a bunch of small palettes that fit together like Lego blocks to create various types of system interfaces. The control palettes I ended up with are of the small, Lego variety. They are designed to be plugged together in several combinations for a mix and match approach to designing interfaces.

Creating the Bank

To get just the functionality I wanted for my Bank control, I took out the navigation system and instead allowed the Bank to show the data for a single Bank only. Mostly because I wasn't concerned with being able to edit Banks, but to provide access to the accounts of a single Bank. Here's the scaled down view of the Bank:

{ewc mvimg, mvimage,!illust.bmp}
Figure 2. The new Bank control.

Adding More Logical Pieces

With a Bank control to provide access to accounts, I added two other palettes that would interconnect with each other and with Bank to form the system. Here's a quick summary of the loose business rules I used.

® A Patron is an entity connected with an account at a Bank.

® A Lender is an entity connected with an account at a Bank that can also loan money to other accounts.

® A Patron can Credit or Debit the balance in its own account as well as Repay a loan to a Lender.

® A Lender can Credit or Debit the balance in its own account and can execute loans to other accounts.

Visibly and operationally, they are relatively simple and neither Patrons nor Lenders are designed to be programmable from code. I haven't bothered with much in the way of security, prohibiting access, or validating payments, other than a simple transaction wrapper around transfers. For example, repaying a loan is simply updating the balances in two accounts, with no checking for whether an overdraft would occur or not. While adding support for these things wouldn't be hard, it is outside the scope of this sample, so it has been left as an exercise for the reader. Here's what the Patron and Lender user interfaces look like:

{ewc mvimg, mvimage,!illust.bmp}
Figure 3. The Patron control.

{ewc mvimg, mvimage,!illust.bmp}
Figure 4. The Lender control.

What Does this Show?

So, what exactly is it that makes this interesting? Well, by designing my controls to connect to and use other controls, a user has to write very little, if any, code to use the pieces as a complete system. For example, the test project that comes with this sample has only this in the way of code:

```
If Not Bank1.Init() Then MsgBox "Didn't Work"
Set Lender1.Bank = Bank1
Lender1.Refresh
Set Patron1.Bank = Bank1
Set Patron1.Lender = Lender1
Patron1.Refresh
Set Patron2.Bank = Bank1
Set Patron2.Lender = Lender1
Patron2.Refresh
Set Patron3.Bank = Bank1
Set Patron3.Lender = Lender1
Patron3.Refresh
```

To create this working sample, I simply dropped three Patrons, a Bank, and a Lender on a form. I set the DataSource and Code of the Bank, the AccountNumber's of the Patrons and Lender, and wrote the 13 lines of code shown above. That gave me a system to manage accounts, and grant or repay loans — a lot of power for very little output.

The real joy is when that inevitable call comes and I need to provide other custom interfaces that use these same business objects. When that day comes, it would be very easy to create whatever combinations were required by simply connecting them, and letting them run. Here are some examples of some combinations that could be put together with just these three controls (adding a few more features to the controls, of course):

® I could use all the pieces in a complete front end for the bank administration network.

® I could use only Patrons on a publicly available Web page that allows users to apply for loans.

® I could use a secure page with only a single Lender on it available to a lending partner of the Bank.

Now do you see what I'm getting at? It's possible because control palettes can work together. Just creating custom ActiveX controls, or even designing control palettes, is not enough any more. Interconnected palettes is what I consider to be the next level in control development.

Take It Away

Now, those of you who are used to reading my articles might wonder that I haven't really shown any code or new techniques in this article. That's because, this particular subject isn't rocket science. Everything that went into the sample, while perhaps interesting, is pretty obvious and doesn't need a lot of explaining. The real trick is thinking about how you build your user interfaces. Connecting even simple controls together is useful but designing control palettes with the ability to interconnect is a powerful technique to master when creating user interfaces.

This article is an extension of the Client/Server Solutions series previously published on the MSDN Library. The series explained some fundamental concepts for doing quality client/server development. A key topic discussed in this series was the Layered Paradigm. This article examines the Layered Paradigm in the light of Microsoft Visual Basic 4.0 and illustrates the mechanics of implementing this approach in a client/server application. The article presents an overview of the Layered Paradigm, lays out an implementation in the service model, and then walks through that implementation. Because this article draws heavily on the fundamentals first laid out in the Client/Server Solutions series, readers are highly encouraged to read that series before reading this document (see "Introduction", below).

This white paper includes the following topics:

Scripting is a technique for embedding program code into Internet documents. While performing its tasks, a transaction processor sends requests for resources and submits results to the server. If the requested resource contains script code, the appropriate scripting engine in either the server or the client evaluates the script as it passes through. Figure 1 shows the resource request path and the locations of the scripting engines.

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. The HTTP resource request path and location of scripting engines.

The script engine on either the browser or the server processes script commands that are embedded in its input stream. Script commands can use the following mechanisms to accomplish their ends:

® Modify the output stream.

® Create objects, procedures, or variables for later use.

® Call procedures, modify variables, set the properties or call the methods of objects provided by other components through COM.

Client-side scripting uses the power of the client computer to provide application services that augment the Hypertext Markup Language (HTML) part of the application. Software components are downloaded as needed and run on the client computer. The client-side script engine has access to these components through COM. A notable addition here is that COM in a distributed environment (DCOM) allows the client computer to use components that are located on computers across a local or wide area network or even across the Internet. By utilizing the computing power of the client, you can reduce the number of requests and the amount of data that is sent over the Internet.

Server-side scripting frees the developer from concerns over client platform differences. Any standard browser can provide basically the same user function when the stream that the client receives contains only standard HTML. The price for this universality is a greater number of client request/server response cycles so your application interface is more apt to be affected by network latency.

Our two applications use data services provided by a COM server developed in Microsoft Visual Basic version 5.0. Using a services model allows us to hide the details of data storage behind an object interface and to increase the security, maintainability, and scalability of our overall solution. See our Corporate Benefits Sample COM servers that are part of Microsoft Visual Studio version 1.0 and the Getting Started articles for a more thorough exploration of these topics. We are using a Microsoft Access .mdb for this sample but the services model would allow us to upgrade to an Open Database Connectivity (ODBC) source without changing the applications. We will also locate the data services server on the Internet server where the server-side application can access it locally through COM and the client-side application can access it across the Internet using DCOM.

CAdmin

Properties	Methods
Banks	Delete
	ExecBoolean
	ExecFillArray
	FillCollection
	FillListBanks
	Init
	Insert
	Term
	Transfer
	Update

CBank

Properties	Methods
Pkid	Refresh
Admin	Term
Accounts	
Address1	
Address2	
City	
Code	
Name	
TypeId	

CAccount

Properties	Methods
Pkid	Refresh
Admin	Term
Bank	
Balance	
Number	
TypeId	

The application displays a list of banks in a vertical frame to the left side of the display. When you select a bank, the system updates the bank detail frame and the account list. You may add new accounts or select an account to edit in the account detail frame. Figure 2 shows the common interface for both applications.

{ewc mvimg, mvimage,!illust.bmp}
Figure 2. Common user interface for both transaction processors.

Client-Side Application Files

Default.htm	Defines and names application frames.
Banks.htm	Creates the bank list.
Empty.htm	Reserves place for dynamically created HTML.
Bankdet.htm	Displays bank detail and creates accounts list.
Detact.htm	Displays account detail.
Detedt.htm	Contains account editing interface.
Detnew.htm	Contains new account interface.

You enter the application through default.htm. The code in default.htm defines names for the application frames so the system can later access variables and objects across frames. The SRC attribute sets the initial Internet resource for the banks frame to banks.htm while it sets the remaining frame source files to empty.htm.

```
<frameset cols="30%,*">
    <frame name="banks" src="banks.htm">
    <frameset name="currentbank" rows="50%,*">
        <frameset name="bankaccounts" cols="45%,*">
            <frame name="accounts" src="empty.htm">
            <frame name="acctdet" src="empty.htm">
        </frameset>
    <frame name="bankdet" src="empty.htm">
    </frameset>
</frameset>
```

When the system first loads banks.htm, the first order of business is to create m_oAdmin of class CAdmin that provides the primary interface for the data services. The m_oAdmin.Init method makes the database connection and FillListBanks prepares the Banks collection for use.

```
<script language="VBScript">
Dim m_oAdmin
Dim m_iCurrBank
Dim m_iCurrAccount
…
Set m_oAdmin = CreateObject("BankSrv.CAdmin")
If m_oAdmin.Init("pathname\banking.mdb") Then m_oAdmin.FillListBanks
```

The next section writes the list of banks to the frame document. Once a script uses document.writeln, the scripting engine ignores any display content in the file and the script assumes responsibility for generating all of the HTML necessary to fill the display. The <a> elements that we write to the frame each contain an index to a Bank, which the system uses to fill the other frames.

```
For iIndex = 1 to m_oAdmin.Banks.Count
    Set oBank = m_oAdmin.Banks(iIndex)
    SBuff = ("<a href=""bankdet.htm"" target=""bankdet"""
    SBuff = sBuff & "onClick=""m_iCurrBank = " & CStr(iIndex) & """""">"
    SBuff = sBuff & oBank.Name & "</a><br>"
    document.writeln(sBuff)
Next
Set oBank = Nothing
 …
</script>
```

When the user clicks on a bank, the onClick handler assigns m_iCurrBank to the selected bank index and the bankdet frame loads bankdet.htm. The following script section from bankdet.htm updates the bankdet and accounts frames according to the selected bank. The system creates a CBank object, m_oBank, and sets it to the selected member of the m_oAdmin Banks collection. The program then walks through the m_oBank

Accounts collection and writes a list of Accounts to the accounts frame.

```
iBank = top.banks.m_iCurrBank
Set m_oBank = top.banks.m_oAdmin.Banks.Item(iBank)
top.accounts.document.open
top.accounts.document.writeln("<strong>Accounts</strong>")
top.accounts.document.writeln("<hr>")
For iIndex = 1 to m_oBank.Accounts.Count
    Set oAcct = m_oBank.Accounts(iIndex)
    sBuff = "<a href=""detact.htm"" target=""acctdet"" "
    sBuff = sBuff & "onClick=""top.banks.m_iCurrAccount = " & CStr(iIndex)
      sBuff = sBuff & """>" & oAcct.Number & "</a><br>"
    top.accounts.document.writeln(sBuff)
    …
Next
Set oAcct = Nothing
top.accounts.document.close
```

The program now writes a bank detail report to the current frame.

```
document.writeln("Bank    - " & m_oBank.Name & "<br>")
document.writeln("Code    - " & m_oBank.Code & "<br>")
document.writeln("Address - " & m_oBank.Address1 & "<br>")
```

When you select an account from the account list, The system sets top.banks.m_iCurrAccount to the selected account index and loads detact.htm into the acctdet frame. Detact.htm includes the following script section that displays account detail as well as Edit and New anchors.

```
Set oAcct = oBank.Accounts.Item(iAccount)
document.writeln("Account Number  - " & oAcct.Number & "<br>")
document.writeln("Account Type  - " & oAcct.TypeID & "<br>")
document.writeln("Account Balance - " & FormatCurrency(oAcct.Balance) & "<br>")
document.writeln("<br>")
document.writeln("<a href=""detedt.htm"">Edit</a><br>")
document.writeln("<a href=""detnew.htm"">New</a><br>")
```

When the user selects the Edit anchor, the system loads detedt.htm. This time, instead of generating the interface elements, the script fills in static HTML elements and provides an On_Click procedure for the Commit button. The text box controls and the Commit button are defined using standard HTML.

```
Account Number<br>
<input type=text Name="AcctNumber"><br>
Account Type<br>
<input type=text Name="AcctTypeId"><br>
<br>
<input type=button value="Commit" OnClick="CommitAccount">
```

The script then fills in the account number and account type.

```
Set m_oAcct = oBank.Accounts.Item(iAccount)
m_oAcct.Refresh
AcctNumber.Value = m_oAcct.Number
AcctTypeId.Value = m_oAcct.TypeID
```

The script defines a subroutine for use when the user clicks on the Commit button. CommitAccount() gets user input from the text box controls and validates the input by attempting to convert it from text to the appropriate data type and monitoring the value of Err.

```
Sub CommitAccount()
…
```

```
Err = False
lNewType = CLng(AcctTypeId.Value)
If Err Then
    MsgBox("Invalid account TypeID.")
    Exit Sub
Else
    m_oAcct.lNewType = lNewType
End if
```

If the input values are valid, pass oAccount, with the new values, to the Admin update method.

```
bResult = m_oAdmin.Update(, m_oAcct)
```

This completes the client-side application, which demonstrates how client-side scripting can work with software components through the client computer in order to provide function beyond what is possible with HTML alone. The server-side version will use the same data services but will access those services through the server's scripting engine and present the browser with a layer of standard HTML.

Server-Side Application Files

Default.asp	Defines and names application frames.
Global.asa	Provides application and session level definitions.
Banks.asp	Creates the bank list.
Empty.asp	Reserves place for dynamically created HTML.
Bankdet.asp	Displays bank detail and creates accounts list.
Detact.asp	Displays account detail.
Detedt.asp	Contains account editing interface.
Commit.asp	Handles account inserts and updates.

The first step toward creating an Active Server Pages (ASP) application in Internet Information Server (IIS) is to create a virtual root directory on the server using the Internet Service Manager. Add the directory to the WWW Service as a virtual Directory with read and execute rights and then set the default document to default.asp. With the virtual root configured, IIS sees the files within the directory as an application and it will provide application and session objects and events. When IIS receives a request for one of the files within the application folder, it manages application and session state and fires the appropriate OnStart events if the session is new. You can create scripts to handle these events and define application and session variables within the reserved file global.asa. The following code from global.asa instantiates and inits the CAdmin object and instantiates CBank and CAccount.

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Session_OnStart
Set Session("m_oAdmin") = Server.CreateObject("BankSrv.CAdmin")
Session("bInitResult") = Session("m_oAdmin").Init("pathname\banking.mdb")
Set Session("m_oBank") = Server.CreateObject("BankSrv.CBank")
Set Session("m_oAccount") = Server.CreateObject("BankSrv.CAccount")
Session("bNewAccount") = False
End Sub
</SCRIPT>
```

Since default.asp contains only HTML definitions for the application framesets and frames, the scripting engine sends it to the browser unchanged. The banks frame requests banks.asp, which contains this next script segment. Note that <% and %> delimit script segments in ASP files.

Refresh the Banks collection of the session-level CAdmin object with the FillListbanks method and write the list of banks to the Response object as HTML anchor tags. The anchors include a bank index in the request to bankdet.asp when they are selected by the user.

```
Session("m_oAdmin").FillListbanks
For iIndex = 1 To Session("m_oAdmin").Banks.Count
    sBuff = "<a href=""bankdet.asp" & "?iBankNdx=" & Cstr(iIndex)
    sBuff = sBuff & " ""target=""bankdet"" >"
    sBuff = sBuff & Session("m_oAdmin").Banks(iIndex).Name & "</a><br>"
    Response.Write(sBuff)
Next
```

In bankdet.asp, the system retrieves the Bank index from the QueryString collection of the Request object and uses it to set the Session level CBank object.

```
iIndex = Cint(Request.QueryString("iBankNdx").Item(1))
Set Session("m_oBank") = Session("m_oAdmin").Banks(iIndex)
```

A quick walk through the accounts provides positive and negative totals for the bank detail report, which the system writes to the Response object.

```
For iIndex = 1 to Session("m_oBank").Accounts.Count
    Set oAcct = Session("m_oBank").Accounts(iIndex)
```

```
    If oAcct.Balance > 0 Then
        cPos = cPos + oAcct.Balance
    Else
        cNeg = cNeg + oAcct.Balance
    End If
Next
Set oAcct = Nothing
Response.Write("Bank    - " & Session("m_oBank").Name & "<br>")
Response.Write("Code    - " & Session("m_oBank").Code & "<br>")
Response.Write("Credits - " & FormatCurrency(cPos) & "<br>")
Response.Write("Debits  - " & FormatCurrency(cNeg) & "<br>")
Response.Write("Net Val - " & FormatCurrency(cPos + cNeg) & "<br>")
```

The script in accounts.asp generates the accounts list using the previously described technique of attaching an index number to an HTML anchor.

```
For iIndex = 1 to Session("m_oBank").Accounts.Count
    Set oAcct = Session("m_oBank").Accounts(iIndex)
    sBuff = "<a href=""detact.asp" & "?iActNdx=" & Cstr(iIndex) & """"
    sBuff = sBuff & " target=""acctdet"" >"
    sBuff = sBuff & oAcct.Number & "</a><br>"
    Response.Write(sBuff)
  Next
```

When you select an account from the account list, detact.asp handles the request and returns the account detail. The script also adds anchors for Edit and New, which are identical except for the way that they set the new account flag by passing QueryString data.

```
iIndex = Cint(Request.QueryString("iActNdx ").Item(1))
Set Session("m_oAccount") = Session("m_oBank").Accounts(iIndex)
Response.Write("Account Number  - " & Session("m_oAccount").Number & "<br>")
…
Response.Write("<a href=""detedt.asp? bNewAccount =0"">Edit</a><br>")
Response.Write("<a href=""detedt.asp? bNewAccount =1"">New</a><br>")
```

When the user selects Edit or New, the request goes to detedt.asp, which stores the NewAccount flag to a session variable and then generates an HTML form that will be posted to commit.asp.

```
Session("bNewAccount") = cBool(Request.QueryString("bNewAccount ").Item(1))
SBuff = "<form name = ""account"" method=""post"" action=""commit.asp"">”
Response.Write(sBuff)
  …
sBuff = "<input type=text Name=""AcctNumber"" "
If Not(Session("bNewAccount")) Then
    sBuff = sBuff & "value=""" & Session("m_oAccount").Number & """"
End If
sBuff = sBuff & "><br>"
Response.Write(sBuff)
…
Response.Write("<input type=submit value=""Commit"" >")
Response.Write("</form>")
```

The system requests commit.asp when a user submits the form with the Commit button. Since this script handles both new and existing accounts, the script creates a new Account object if necessary.

```
If Session("bNewAccount") Then
    Set Session("m_oAccount") = Server.CreateObject("BankSrv.CAccount")
    Set Session("m_oAccount").Admin = Session("m_oAdmin")
    Set Session("m_oAccount").Bank = Session("m_obank")
```

```
End if
```

The system retrieves input parameters from the Request.Form object and applies them to the account object.

```
sNewNumber = Cstr(Request.Form("AcctNumber"))
Session("m_oAccount").Number = sNewNumber
lNewType = CLng(request.Form("AcctTypeId"))
Session("m_oAccount").TypeID = lNewType
```

Finally, the system either inserts or updates the Account object based the value of NewAccount.

```
If Session("bNewAccount") Then
    Result = Session("m_oAdmin").Insert(, Session("m_oAccount"))
Else
    Result = Session("m_oAdmin").Update(, Session("m_oAccount"))
End If
```

The data services component, as well as the database file, is located on the server for both applications so we can perform a reasonable comparison between client-side and server-side scripting. See the Corporate Benefits Sample COM servers and the documentation included with Visual Studio version 1.0 for more expansive coverage of deployment issues.

Create the registry entries necessary for remote registration on your client machines by importing banking.reg with regedit.exe. Note that these registry entries include the SafeToScript and SafeToInitialize flags under implemented categories that you will have to create by hand if you register the servers using another method.

Use racmgr.exe, included with Visual Basic version 5.0, to configure the server on your clients as well as the server. Configure the data services classes for launching on the server via DCOM. Specify the server name if you are running on an intranet or the IP address of the server if you are running over the Internet.

Finally, use dcomcnfg.exe to configure your Windows NT version 4.0 computers for DCOM security and launch permissions. The easiest way to get started with DCOM security is to open default security as wide as possible to get the system running, then make it progressively more restrictive until you have the desired level of security. Start with a default authentication level of none and a default impersonation level of anonymous. With default security configured, configure each data services class to use default security.

Although I've presented client-side and server-side scripting as distinct techniques that represent opposing design philosophies, they are not mutually exclusive. In fact, they work best when you use them together, along with ActiveX controls and components. In future projects, I'll cover the issues that stand between these introductory samples and real applications, and I'll explore related Internet and component software issues of interest to client/server developers.

This is an example of a user service, not a business service.

For more information, see The Services Paradigm.

This is an example of a data service, not a business service.

For more information, see The Services Paradigm.

This is an example of a data service, not a business service.

For more information, see The Services Paradigm.

This is an example of a business service.

For more information, see The Services Paradigm.

User services provide the application with its graphical interface.

For more information, see The Services Paradigm.

Business services implement the business rules for an application.

For more information, see The Services Paradigm.

Data services provide low-level manipulation of data in the database.

For more information, see The Services Paradigm.

The Web Developer is responsible for adding server-side script to Active Server Pages. The script invokes components and controls that are created by the Programmer.

For more information, see Web Developer.

The Programmer creates ActiveX server components. The Web Developer is responsible for adding server-side script to Active Server Pages. The script invokes ActiveX server components.

For more information, see Web Developer.

The HTML Author creates the content in a Web site by adding HTML tags and hyperlinks.

For more information, see Web Developer.

The Graphic Artist creates images that are included in Web pages.

For more information, see Web Developer.

The Web Developer defines the Web site architecture.

For more information, see Web Developer.

The Web Developer defines the Web site architecture.

For more information, see Web Developer.

The Web Developer defines the Web site architecture.

For more information, see Web Developer.

The Web Developer defines the Web site architecture.

For more information, see Web Developer.

Visual InterDev is a tool that you use to create HTML pages and Active Server Pages for Web sites.

For more information, see Programmer.

Visual Basic is a programming tool that you can use to create ActiveX server components and ActiveX controls.

For more information, see Programmer.

Microsoft Transaction Server is not a programming tool. Microsoft Transaction Server is an application that provides transaction and resource management for ActiveX server components.

For more information, see Programmer.

Microsoft SQL Server is not a programming tool. Microsoft SQL Server is a database management system.

For more information, see Programmer.

Visual InterDev is a tool that you use to create and manage the HTML pages and Active Server Pages in a Web site.

For more information, see Programmer.

Visual Basic is a programming tool that you can use to create ActiveX server components.

For more information, see Programmer.

Microsoft Transaction Server provides transaction and resource management for ActiveX server components.

For more information, see Programmer.

Microsoft SQL Server is a database management system that you can use to create and manage databases.

For more information, see Programmer.

Wouldn’t it be nice to drag a control onto a form, set a few properties or write a few lines of initialization code, and have an instant interface to your data? Control palettes allow you to have exactly this type of functionality. Wouldn’t it be even better still to have a solid, multilayered, client/server application wrapped up into a control? Anyone can throw a few controls together, bind them to a database, and compile the whole enchilada into an ActiveX control. Will it be robust? Extensible? Programmable? Will it adhere to a solid design methodology? My money is on none of the above. After reading this article, however, you'll be able to build a control you could take home to Mom.

Our design methodology breaks out the control’s functionality into four separate, yet inter-dependent, layers.

The User-Interface Layer

The user-interface layer is what you will show the user. Controls that display data and navigational routines belong here. No other layer depends on the user-interface layer; however, the user-interface is directly dependent on the data layer.

The Data Layer

All of the control's data is contained and manipulated here. The data layer supports the interface and depends on the transaction layer.

The Transaction Layer

The transaction layer is where all of the transactional elements are assembled, such as insert and update routines; however, they are not executed here, that comes next. This layer supports the data layer and depends on the external access layer.

The External Access Layer

The external access layer performs your transactions on an external data source. This layer directly supports the transaction layer and depends on nothing.

Often when given a design model, we do our best to modify our application needs to fit the model. This type of rigidity only causes headaches and can lead to code hacks and poor performance. The layered paradigm does not propose stringent separation between the layers. In fact the lines between each layer are extremely fluid. Figure 1 depicts a purist approach to coding for the layered model.

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. Purist implementation.

From a purist perspective, the user-interface only calls the data layer, which in turn only calls on the transaction layer, which talks to the external access layer. In real-world applications, this type of strict enforcement of the model may not be practical. Notice, in Figure 1, the redundant calls to insert a record — an Add method calls an Insert() service in the transaction layer. Figure 1 also shows redundant calls to update, delete, and run any other transaction services available. The code is growing and growing and performance is fading away with each extra step. That's just one drawback. You'll also have to keep your naming conventions straight, maintenance will be a bear, and any hope of reuse diminishes with each added service.

When designing an application using the layered paradigm, it's not important that your application show a physical separation of code denoting the different layers. What is important is the dependencies of code from one layer to another. Understand that the layered paradigm is a logical model, not a physical one. The BankView control palette built in this article exhibits each layer and the correct dependencies. Figure 2 depicts a more realistic implementation of the layered paradigm.

{ewc mvimg, mvimage,!illust.bmp}
Figure 2. Realistic implementation.

It’s important to decide up front how you plan to implement the layered methodology. If your application includes many data objects that are going to need transaction services, then the implementation used in BankView may not be the best approach.

A Few Approaches

As you will see, BankView uses transaction functions that accept optional data object parameters. The transaction function then identifies the object and builds a transaction specific to that object’s type. This is great for the two data objects of the BankView control, but what if your application has twenty or more data objects? Imagine the size of the function that accepts twenty optional parameters and creates a transaction string for each! In this instance it may be better to build your transaction building code — not external access — into your individual data classes. This method still fits the model because the transaction lines are fluid; again this is a logical model, not a physical one. It’s the dependencies that matter, not necessarily the location or call structure of the code. An even better approach is to give your data classes a .Deleted property and a Persist() method, because it could scale up to handle many data objects. A simple scenario would have the Persist() method acting in support of the transaction layer. Persist() would determine if and how the object would persist (by examining its .Deleted property, or checking to see if a primary key exists yet) and then handing off the object to the proper service of the transaction layer.

Object Model Ramifications

It is also important to consider how your implementation affects your application’s object model. You don’t want to end up with a model that is difficult or clumsy to use. For instance, in BankView’s implementation of layers, physically storing a data object (adds, edits, or deletes), AKA persistence, requires that the object be delivered directly to a transaction service, for example oAdmin.Delete(oBank). The purist approach described above requires a method call on the object specific to the transaction, oBank.Remove(). The final scenario described above requires a single call for any transaction, oBank.Persist (if this object were being removed, you would need to set the .Deleted property).

At first glance, none of the scenarios look too challenging to use. But what if you had several more types of transactions? BankView’s model and the purist model become a lot more difficult to manage. A .Persist method maintains its ease of use. Of course, now the complexity of coding the .Persist method comes into play. It’s a game of tradeoffs.

When using the layered paradigm as a model for your applications, it’s important to make implementation decisions up front. Ease of development, ease of use, reusability, flexibility, performance — all need to be considered. Spend time planning; it could be very difficult to implement changes later.

The BankView control palette exposes an interface detailing bank and account information. Users can navigate through bank and account data as well as add, edit, and delete data. Next, you'll see how the parts all fit together.

BankView: User Interface

The BankView user interface includes all the visual elements of the control and includes the methods and properties shown in Figure 3.

{ewc mvimg, mvimage,!illust.bmp}
Figure 3. BankView interface, methods, and properties.

The layered model states that the user interface is totally dependent on the data layer. Does the interface reflect that? Examine the following code from the .DisplayBank method of the control.

```
With oBank
    txtBank(icBankName) = .Name
    txtBank(icBankCode) = .Code
    txtBank(icAddress1) = .Address1
    txtBank(icAddress2) = .Address2
    txtBank(icCity) = .City
    . . .
    FindItem lstBankType, .TypeId

    For Each oAcct In .Accounts
       lstAcct.AddItem Trim$(oAcct.Number) & _
                   " " & Format$(oAcct.Balance, _
                            "Currency")
    . . .
   Next oAcct
    . . .
End With
```

Notice that the user-interface's only contact with data is by referencing the data class CBank. Without the data class, the interface is empty and non-functional. The user-interface does indeed adhere to the layered model.

BankView: Data

The CBank and CAccount classes contain the data used by the BankView interface. Our object model is a simple one-to-many relationship between Bank and Account. A single Admin object is referenced by each object to handle transaction and external access requests. Figure 4 depicts the object model.

To clarify, an Admin object is only created once for the whole model. Each Bank or Account object then sets a reference to that object. In the BankView control, you'll see that each oBank and oAccount object sets a reference to m_oAdmin.

{ewc mvimg, mvimage,!illust.bmp}
Figure 4. Object model for the BankView Control Palette.

The data classes fit nicely into the data layer since they act only as a container for data. Data isn't sent anywhere, the user interface must reference it. The data objects don't get the data from a database; instead, data must be requested from another layer. The code below illustrates how data gets from text box controls on the interface to the bank data object, and then to the transaction layer function, Insert.

```
Set oBank = New CBank
With oBank
    .Name = txtBank(icBankName)
    .Code = txtBank(icBankCode)
    .Address1 = txtBank(icAddress1)
    .Address2 = txtBank(icAddress2)
    .City = txtBank(icCity)
```

```
    .TypeId = lstBankType.ItemData(lstBankType.ListIndex)
End With
If m_oAdmin.Insert(oBank:=oBank) Then
. . .
```

As you can see, the bank object doesn't know how to insert a new bank, it merely contains the data that needs to be inserted. Therefore, the bank object must be sent to the Insert() function, which resides in the transaction layer. Since the data classes directly support the interface and depend upon the support of the transaction layer, they fit nicely into the data layer of the model.

BankView: Transaction

The transaction layer of the BankView control includes the following functionality:

GetDomain()	Builds a transaction to fill a collection with domain values (for example, Bank Insurers).
Insert()	Builds a transaction to add the data in the data object to the data source.
Update()	Builds a transaction to update a record in the data source with the values of a data object.
Delete()	Builds a transaction to delete a record in the data source.

Examine the following code from the Insert() subroutine.

```
With oBank
. . .
    Set cParms = New Collection
   cParms.Add Trim$(Left$(.Address1, 60)), _
                    "[@Address1]"
   cParms.Add Trim$(Left$(.Address2, 60)), _
                     "[@Address2]"
. . .
cParms.Add .TypeId, "[@BankTypeId]"
End With
If Not ExecBoolean("pcIns_Bank", cParms) Then
. . .
```

The Microsoft Access database contains QueryDefs with parameter variables. Insert() prepares a transaction by storing the object data and associated QueryDef parameters in the collection cParms. The parameter collection and the name of the QueryDef that is to be executed (in this case, pc_InsBank), is then delivered to the external access layer for handling — courtesy of the ExecBoolean() function. The functionality of the transaction layer should mostly be reusable beyond this component.

BankView's transaction layer also follows the layered approach since it supports the transactional needs of the data classes and is dependent upon the functionality of the external access layer.

BankView: External Access

Finally, you can see a description of the layer that actually operates on a data source in conjunction with the requests of the transaction layer. BankView's external access layer includes the functions ExecFillArray() and ExecBoolean(). Examine this snippet from the ExecBoolean() function.

```
m_wsAccess.BeginTrans
Set qryDef = m_dbAccess.QueryDefs(sQry)
With qryDef
    For Each pParm In .Parameters
        pParm.Value = cParms(pParm.Name)
    Next pParm
    Err = False
```

```
    .Execute dbFailOnError + dbSeeChanges
    If CBool(Err) Then
        m_wsAccess.Rollback
. . .
```

ExecBoolean() is designed to operate against a specific type of a data source. In this case, it operates against a Microsoft Access database. ExecBoolean() takes the transaction parameters assembled by the transaction layer, as well as the name of a QueryDef that will handle the transaction, and executes it against a Microsoft Access database file. The functions of the external access layer should be completely reusable when used against the data source for which they were designed.

The external access layer of the control completes the layered model. It supports the requests of the transaction layer and depends upon nothing.

The layered paradigm is an atomic design model in that it breaks functionality down into related groups. Because functionality of the BankView Control Palette has been segregated into atomic units, functionality changes shouldn't be catastrophic to the control.

Say you've built the Control Palette and it's been in production for a while. Because of the amazing popularity of the control, your Microsoft Access database is approaching its performance limits. You need to change from a Microsoft Access back end to a SQL Server back end. Is it back to the drawing board for the control? Not at all. Are you going to have to gut major sections of your code wherever you've referenced a Microsoft Access database? No way. Because of the layered design, the external access layer of the control contains all of the database-specific code. You could rewrite ExecBoolean() as ExecBooleanSQL() to act on SQL Server data; or better yet, you could get the best of both worlds by including it as ExecBooleanMDB() and ExecBooleanSQL(). Now the external access layer (and thus, the Control Palette) can act upon both Microsoft Access and SQL Server data sources. That is one of the beauties of layered design — transport independence.

Okay, you've built the darned thing. Now how do you go about using it? Using the BankView control palette is as easy as dropping it onto a form and setting its DataSource property to point to your database. Run your project and you'll have a ready-made data interface.

Coding Control Operations

You can also control the control palette through code. You can turn off the navigational controls and navigate the Control Palette's data using the control's methods. The following code snippet details how you can initialize and navigate the BankView control using code.

```
Public Sub Test()
Dim sDBPath as String
sDBPath = "c:\work\banks.mdb"
BankView1.NavStyle = CodeNavigate
If Not BankView1.Init(sDBPath) Then
       MsgBox "Didn't work."
    Exit Sub
End If
BankView1.MoveLast
BankView1.RemoveBank
BankView1.MoveLast
End Sub
```

As you can see, the control palette doesn't require a Ph.D. to operate it using code.

Okay, you've designed it, you've built it, and you've even seen it run in a tiny, sample Visual Basic project. So what will you do with a canned interface into your bank's data source, or any control palette for that matter?

What you have created is the ultimate in code reusability. After creating and testing the control, what you wind up with is a bona fide ActiveX control! You can drop this control into a Visual C++ application . . . you can drop it into a Visual FoxPro application . . . hey, even another Visual Basic application. In fact you can drop this control into any environment that accepts ActiveX controls. You could also drop this control onto an HTML page and, in seconds, have a Web application! Is it starting to sink in?

Imagine having a toolbox full of control palettes! Building applications becomes nothing more than dropping your controls onto forms and writing any necessary administrative code.

Step back and examine the material you've just read. Notice how much of the article was dedicated strictly to design issues and considerations. It’s important to understand up front how you are going to design your applications. Spend time on your design, you will save yourself many headaches later. Of course, I’m sure that our readers would *never* use a design-as-you-go technique, but there are folks who do . . .

If you'd like to read more about the Layered Paradigm, be sure to read Ken Bergman's "Client/Server Solutions" series of articles, listed below. (The series is also available on the Jan 1997 MSDN Library):

® The Architecture Process

® The Design Process

® The Basics

® Coding Guidelines

® Implementing the Layered Paradigm

® Cursors, Asynchronous Queries, and Handling Multiple Result Sets

® Leveraging SQL Server Services in a Transaction Processing Environment

The source code for the BankView control palette can be found on the *Developer Network News* Online at http://www.microsoft.com/devnews/. It is meant to be used as a study tool and should by no means be considered production code. By all means, though, get the sample code and play with it. Perhaps you can come up with a more flexible and reusable implementation of BankView that utilizes the layered paradigm.

Watch this space for future implementations of layered development. Click here to connect to the *Developer Network News* Online Web site.
{ewc mvimg, mvimage,!intjump.bmp}

The **Command** and **Recordset** objects can also be created with the **CreateObject** method.

For more information, see ADO Object Model.

The **Recordset** object can also be created with the **CreateObject** method.

For more information, see ADO Object Model.

The **Connection** object can also be created with the **CreateObject** method.

For more information, see ADO Object Model.

The objects **Connection**, **Command**, and **Recordset** can all be created with the **CreateObject** method.

For more information, see ADO Object Model.

There is not an **OpenRecordset** method for the **Connection** object.

For more information, see Retrieving Records.

There is not an **OpenRecordset** method for the **Recordset** object.

For more information, see Retrieving Records.

To retrieve the records from the Students table, you can call the **Open** method on the **Recordset** object, and pass a query and the **Connection** object as arguments.

For more information, see Retrieving Records.

You cannot pass an SQL string as an argument to the **CreateObject** method.

For more information, see Retrieving Records.

When an error occurs in an Active Server Page that does not have an error handler, processing stops and the error message from the database is included at the bottom of the Web page returned to the user.

For more information, see Handling Errors.

In addition to the error message, the Web page will include any HTML text that was generated before the error occurred.

For more information, see Handling Errors.

The Web browser is not aware of any processing errors that occur on the Web server. The server places the error in the HTML response, and returns it to the user.

For more information, see Handling Errors.

When an error occurs in an Active Server Page that does not have an error handler, processing stops. The error description and any HTML text written before the error occurred is included in the Web page returned to the user.

For more information, see Handling Errors.

By caching records on the client, the user can scroll through the records without requesting another page from the server.

For more information, see Using the Advanced Data Connector.

Data is cached on the client, not in a business object.

In general, business objects do not cache data. This would not benefit the user, because extra trips to the server would still need to be made to retrieve the cached data from the business object.

For more information, see Using the Advanced Data Connector.

Data is cached on the client, not the server.

Caching data on a server would not benefit the user because extra trips to the server would still need to be made to retrieve the data. As more users were added to the system, server resources would be consumed by storing cached data.

For more information, see Using the Advanced Data Connector.

Data is cached on the client, not the database server.

Caching data on a database server would not benefit the user because extra trips to the server would still need to be made to retrieve the data. As more users were added to the system, server resources would be consumed by storing cached data.

For more information, see Using the Advanced Data Connector.

The BINDINGS attribute of the **AdvancedDataControl** object stores the names of all data-bound controls to bind to. Once set, these controls will be filled with data from the recordset in the **AdvancedDataControl** object.

For more information, see Inserting the Advanced DataControl Object.

You set the DATASOURCE attribute when working with the **Data** control in a Visual Basic application. The Advanced Data Connector (ADC) does not use this control for Web pages.

For more information, see Inserting the Advanced DataControl Object.

Data-bound controls do not have a BINDINGS attribute.

For more information, see Inserting the Advanced DataControl Object.

It is not possible to access data in the **AdvancedDataControl** object by using client-side script.

For more information, see Inserting the Advanced DataControl Object.

Quite a long time ago, I wrote a series of articles on client/server development from an enterprise point of view. One of the main topics I discussed was the Layered Paradigm, a perspective for implementing the standard components of a client/server application. Of course, because this was quite a while ago, the implementations were affected by the limitations of the previous versions of Microsoft Visual Basic. With the new capabilities in programming tools, the Layered Paradigm becomes not only easier to implement, but also more crucial to success than ever. But I'm getting ahead of myself. Let me explain what this article will cover. First, I'll give a brief overview of the Layered Paradigm, then I'll talk about the background of this article, and finally I'll illustrate an object-based implementation under Visual Basic 4.0. I'll go through the various layers, show some code, explain some of the attributes and limitations, and then discuss some alternatives at each layer. I'll finish up with a short discussion of how the paradigm fits into other aspects of enterprise development, including workbenches and shared code.

Because this article draws heavily on background information from the Client/Server Solutions series previously published in the MSDN Library, you should first read these four articles in the series:

® The Architecture Process

® The Design Process

® The Basics

® Coding Guidelines

I first started preaching about the Layered Paradigm when almost all solutions were considered to be two-tiered, or basic client and server. The application lived on the workstation and the data for the application was shared on a server. I implemented quite a few of these and several are still in use today at the enterprise level. Back then, the idea of a really advanced system was for it to be three-tiered, meaning that there were three physical machines that made up the system: the workstation applications (or clients), the server, and a piece in the middle that managed the relationship between the clients and the server. This piece supposedly was to wrap the "business rules," whatever the heck those are.

Of course, we all had businesses to run, so rarely did anyone get around to developing a three-tiered system. I usually had to make do with two-tiered systems and good programming techniques. That's where the Layered Paradigm came from. By "layering" or modularizing the functional components of a traditional system, the challenges of creating client/server systems could be addressed one at a time instead of all at once. This "layering" or modularization was *very* conducive to code reuse, sharing, and so on. Of course, even with these benefits, it still took a while to catch on. It was a new idea, and in those days, a logical modularization of an application was uncommon, and there certainly wasn't a term for it.

Skipping forward to today's world, you can now hear the term *three-tiered* used in several ways. Some still use it in the traditional (or pure) sense when referring to a system that has three physical components, none of which is local to the other components. There is also a new use for the term. It can now be used to refer to three logical distinctions *within an application*. These logical sections are usually referred to as the user interface, the business rules, and the data sections. It's important to understand the varieties in the terminology in order not to confuse this perspective with the layers of the Layered Paradigm. There is a good way to tell the difference: When we talk about three-tiered *applications* logically, there are still only three tiers, but in talking about *any* application from a layered perspective, there are always *four* layers.

Here are the four layers:

- *User interface.* This is the topmost layer and is where all user interaction is completed. It supports no dependents and is directly dependent on the data interface. The user interface layer is almost entirely nonreusable and is completely contained within the application boundary.
- *Data interface*. This is where all data is contained, or manipulated, in memory. The data interface layer directly supports the user interface and is directly dependent on the transaction interface. This layer is mostly nonreusable and is usually contained completely within the application boundary. However, in the case of distributed reusable *component areas*, the component might manage its own data. In terms of location, this layer is usually on the client (or workstation), but location is not part of its definition.
- *Transaction interface*. This layer is where all transaction-based processing of data occurs. It *coordinates* all permanent storage of data, either through file or database access. The transaction interface layer supports the data interface and is dependent on the external access interface. This layer may be reusable and may extend past the application boundary, in the same way as the data interface. It is also important to note that the definition of this layer does not suppose any physical location boundary. It can just as easily be moved between the workstation and the server, or any location in between.
- *External access interface*. This layer is responsible for all the communication between an application and external data sources. External data sources may be in such forms as files, databases, hardware, and so on. The external access interface layer supports the transaction interface and in most cases should be completely reusable. It might also be managed strictly by a *component area* that falls outside the component boundary. Once again, this layer has no specific location associated with it, although for performance, it will most likely reside in the same location as the transaction interface.

To return again to the distinction between the Layered Paradigm and three-tiered logical modeling, the following stacked diagram might help you understand why the Layered Paradigm has matured to four layers.

{ewc mvimg, mvimage,!illust.bmp}
Figure 1. The four layers of the Layered Paradigm

The figure shows how the Layered Paradigm draws specific lines over who owns what data at each step in the process. The Layered Paradigm also makes it apparent where each type of operation takes place in the process. For example, in a three-tiered application, there is no room to model the translation between the requests for a particular type of database operation and the specifics of that operation for a particular database or transport. Because of this limitation, many applications today lack the ability to easily change the database or transport for which they were originally implemented. In an application implemented using the Layered Paradigm, only the bottom two interfaces can *ever* be affected by changing the database or transport. In most cases, it will only be the external access interface. Only in extreme cases will the transaction interface require

modifications.

To avoid any confusion at this point, let me make very clear a distinction between the transaction interface and the data interface. The data interface simply provides common mechanisms for both storing data outside the database (that is, in memory, file caches, and the like) and for the applications to access this data. The transaction interface never owns the data; it is simply an operator interface. The transaction interface just passes data from a specific external access interface to the data interface; when the time comes for updates to the data, the data interface gives the transaction interface the data to be operated on and tells it what type of operation to affect. So, the transaction interface never owns data — it only operates on data. The data interface never operates on data — it only owns it. This distinction can be very important, and in later sections I'll explain why.

So, now that you know a little about the evolution of the Layered Paradigm, let's move right into an implementation.

Before I start flinging code around, it's important to outline the specifics of the implementation so that the modularization and ownership boundaries are defined. In an implementation of the Layered Paradigm, the ownership lines are *very* fluid. Therein lies its greatest advantage. Of course, whenever you are working with a technique that can be very subjective or fluid, it's important to outline the assumptions you are working under so that the decisions you make as you progress will have a solid grounding. Models work very well for this type of thing, so that's how I'll explain the architecture of this particular implementation. By the way, I call this a *service model* implementation because it is really just a collection of system services.

The following diagram lays out the components that are required for an implementation of the Layered Paradigm to work. The details of the components are irrelevant. The key here is to understand the services that each layer provides to an application. A good way to understand this is to think of the various tasks that must be done frequently when developing client/server or transaction processing systems. Issues like logging and generically accessing data are among the things that should come to mind. If you are having a hard time understanding services as they relate to client/server or transaction processing systems, it may help to give the diagram a quick once-over for now and then refer back to it as I discuss the components in more detail later.

{ewc mvimg, mvimage,!illust.bmp}
Figure 2. The required components in a Layered Paradigm implementation

Okay, now that you have a rough idea of the types of services that our implementation will have, let's look at them in depth and start to figure out how an application might use them.

This section includes the following topics:

The log service is on the bottom of the interface chain, so I'll start with it. Essentially, it will have the ability to take an item (whatever it might be) and post it, presumably to some type of persistent storage mechanism such as a file or database. For ease of implementation, we are going to assume that log services are noncritical so that we can write no-fail code in this service. As such, the code can simply be a method of the log service with no return value or a type of fire-and-forget routine.

Since I can envision several types of log mechanisms (different files) for storing my log data, I probably want to provide some standard flavors so that developers have less subclassing to do when they actually want to use my service in an application. I'll just be arbitrary and say three flavors. The first log mechanism is strictly for errors, the second is strictly for database transactions, and the third is a catch-all for the other two and anything else as well. The third is a kind of a super log, though maybe with less specific information. Here's how I would expect to use this service, and what I would get:

```
Log.PostItem NOT_ERROR, "clsInterests.Prepare", STR_MSG_ENTER
```

I'm using an object called **Log** and executing the **PostItem** method. Presumably I have some constants defined. The first is NOT_ERROR or some numeric code that indicates that this item is informational. Notice that the Log object I am using actually dictates which Log the item goes to, and the NOT_ERROR code in this case simply tells me that the item is informational. In an error log, this might be a severity level or the error number. I then give the class and current method name, or whatever process-identifying string I want. Then I give the details of the item. In this case, it's the constant STR_MSG_ENTER, which is presumably a string that signifies that the code is entering a new procedure.

Here is an example information log:

18 Mar 96 17:03:42	GetVersion	EXEC pcGet_Version 'MyApp'
18 Mar 96 17:03:43	mpDbsJet.ExecFillCollect	EXEC pcGet_Version 'MyApp'
18 Mar 96 17:03:43	GetPermissions	EXEC pcGet_Permission 'MyApp', 'kenbe'
18 Mar 96 17:03:43	CmpDbsJet.ExecFill Collect	EXEC pcGet_'MyApp', 'kenbe'
18 Mar 96 17:03:44	GetRowLimit	EXEC pcGet_REGS "Options", "Row Limit"
18 Mar 96 17:03:44	CmpDbsJet.ExecGetText	EXEC pcGet_REGS "Options", "Row Limit"
18 Mar 96 17:03:45	Msg.Retrieve	Enter
18 Mar 96 17:03:45	Msg.Retrieve	EXEC pcRet_MAILPERS 'kenbe'
18 Mar 96 17:03:45	CmpDbsJet.ExecFillList View	EXEC pcRet_MAILPERS 'kenbe'
18 Mar 96 17:03:46	Msg.Retrieve	Exit

As you look at the log extract, notice that some information is repeated. This is because each of the layers can operate more or less independently, and in the case of log services, they all make use of the same services. Having just one service that all the layers can talk to makes it easy to centralize issues such as formatting or the type of persistent storage. Perhaps we want to begin storing the log information into a local database instead of a text file. Simply modifying this common service will enact the changes across the system.

The error-handling service is another very common service that will need to be accessible from just about every part of the system. The error-handling service is tied closely with the error flavor of the log services. Essentially, the developer would only require one step to handle an error and to log it in all required logs.

By "handle an error," I mean to store the information about an error for persistence. Programming in this way requires some assumptions, but once they are fully known and understood, the load for all developers in the system is considerably lessened. I'm first going to just charge in and lay the assumptions out. After I've talked about each of the assumptions, I'll explain why they depend on each other and why they should be made.

The first major assumption is that operation procedures that don't return data should not return error codes, only Boolean values. If failure conditions are allowed, the procedure must store error information using a service and return a False. In other words, all routines that do work and from which code might branch depending on a return value, should only return Boolean values. Functions can return data or Boolean values, but not error codes. I'll explain this more in a bit.

The second assumption is that noncritical code should be in **Sub** procedure calls whenever possible. If the operation being done is not critical to the overall success of a higher operation, then it should be in a procedure, not in a function. Someone calling into noncritical code should not have to worry about making branching decisions based on returns from noncritical code. Again, this can be abstract and it will become clearer later, when I bring this all together.

The third assumption is that noncritical sections must not require their caller routines to check for or respond to error conditions. In other words, if you decide to have a Boolean function to conduct a portion of a larger process, the larger process does not have to deal with errors returned by the Boolean function. The function handles the error itself, up to and including preparing messages for the user and storing information in a log. This assumption can be hard to understand, but it gets clearer when considered with the other two.

When you combine these assumptions, it means that the interfaces for services should usually be **Sub** procedures and only occasionally require Boolean function calls. When they are **Sub** procedures, they cannot require that the caller check the error handler for possible error codes. Now let me explain a little about why I can say these things.

See, the first two assumptions simply define standard interfaces between worker functions and critical code. Once this distinction has been made, we further limit the overlap of responsibilities between types of code. If the calling code does not have to check for and handle error conditions, the code in the worker functions can be made to insulate the caller even more from the details of implementation. At this level of encapsulation, we are not only separating the interfaces, we are requiring that the worker functions be able to operate independently of each other. Replacing dependencies in this manner is widely known to be one of the keys to making reuse practical.

This structuring and classification of code brings some added benefits, because when these concepts are actually put in practice on a project, the impact of different styles of coding on a particular application can be lessened. Also, by requiring a high level of independence in code, we can make the code for an application much more compact and much less error prone. In the end, all these benefits mean that the code you end up with is less error prone and many times more reusable.

Here's a small example of coding using the assumptions:

```
'This routine deletes the current person.
Private Sub Delete()
On Error Resume Next
Dim sParm   As String

'Do preliminary validation.
If Not CBool(lPKId) Then Exit Sub
Screen.MousePointer = vbHourglass
SysCon.StatusSet STR_STS_DEL & sMe
Log.PostItem NO_ERROR, "clsPerson.Delete", STR_MSG_ENTR
'Build the parameters list.
'With only one field this could be done inline,
'but that wouldn't be consistent with other parameter-building techniques.
sParm = CStr(lPKId)
```

```
Log.PostItem NO_ERROR, "clsPerson.Delete", qry_PrsDel & sParm
'Call Boolean execution function; it works or it doesn't.
'If it doesn't, the error handler will know why.
'I just tell it to tell the user.
If Not DBSvc.ExecBool(DS("MAIN"), qry_PrsDel & sParm) Then
    'Ask the Error Service to display the last error.
    'Stick standard could not delete message in front, and
    'standard contact system administrator message on the end.
    SysCon.ErrSvc.Display vbCritical, STR_MSG_NODEL, STR_MSG_CNTSA
End If
'Private retrieve function to reload data
Retrieve
Log.PostItem NO_ERROR, "clsPerson.Delete", STR_MSG_EXIT
SysCon.StatusReset
Screen.MousePointer = vbDefault
End Sub
```

This code is easy to read and very compact. Notice the use of the subroutines to handle all of the noncritical tasks, such as logging and setting the status indicators. There is literally one line of error handling in the whole routine! All the **Sub** procedures will handle and log their own errors and activities. Even the database procedure is straightforward. Either it returns or it doesn't. If it doesn't, it simply asks the error-handling service to tell the user about the last error. There is no cleaning up — all of the dirty work is done by the error-handling service.

The data controller services operate on many of the same assumptions as the services I've already discussed. They encapsulate every type of database operation that an application will ever need to perform. They do this in a generic way so that the transports can be interchangeable. For example, I could implement a version of the service that used Microsoft Jet to communicate with databases. I could implement another version that used Remote Data Objects (RDO) to accomplish the same tasks while using the same interface. I could go even further and implement a version that used sockets to talk across a network to a service provider that operated against flat files or a mainframe. As long as I can implement the same interface at this level, I have complete transport independence for my application and its services.

In deciding what the implementation actually looks like, it is important to consider the entire transaction model and match the service interface to it. For more information about designing and implementing transaction models, watch for an article later in the Client/Server Solutions series. For now, here is an example interface that I have implemented. The operator class interface looks like this:

```
'Interface to data service class that is an operator class.

'ds As CmpDbsJet.Source is the data class for data services.
'An instance of CmpDbsJet.Source holds information about what a database looks
like.
'The data services can use the contents of a CmpDbsJet.Source to know
'how to operate. A caller passes an object of type CmpDbsJet.Source in to
'any operator function and the function operates on the object passed in.

'Private Function CheckConn(ds As CmpDbsJet.Source) As Boolean
'Public Function ExecBool(ds As CmpDbsJet.Source, ByVal sQry As String) _
                         As Boolean
'Public Function ExecFillArray(ds As CmpDbsJet.Source, ByVal sQry As String, _
'                              sArray() As String) As Long
'Public Function ExecFillCollect(ds As CmpDbsJet.Source, ByVal sQry As String, _
'                                cTmp As Collection) As Boolean
'Public Sub ExecFillLBItem(ds As CmpDbsJet.Source, ctlMe, ByVal sQry As String)
'Public Function ExecGetText(ds As CmpDbsJet.Source, ByVal sQry As String)_
                            As String
'Public Function Init(ds As CmpDbsJet.Source, objE As objErrHnd, _
'                     objILog As objLogEvent, objTLog As objLogTrans) As Boolean
'Public Sub Term(ds As CmpDbsJet.Source)
```

Note that I am using an **operator** class and a separate **data** class. A data class has no public methods; it only serves to hold data, in much the same way a user-defined type or collection might. A big advantage is that it can have code that executes when the properties of the class are set. This is useful when you need an entity to hold data but need to require that the data take a specific format or be in a particular order. However, an operator class doesn't really own any data. It operates on a data class that is passed into it. This is useful when you have operations that are similar but need to operate on different classes. For example, you might have two entities in your system, hourly employees and salaried employees. Both need to be taxed using the same algorithms. To do this, you code an operator class to do the taxing by simply passing in the object currently being evaluated. Because the two objects share some of the same properties, the operator can perform its function equally well on both. In the example interface shown above, it is the operator class that is shown. After all, there really isn't much to the interface of a data class.

In this particular implementation, I envisioned the possibility of having the application require access to more than one database at a time. Therefore, the databases can be accessed using a key to a collection of data classes that hold the information about the particular database. For example, you could establish the published key for the database that holds tax information as "TAXDB." This key enables a user to find the right object of type **CmpDbsJet.Source** in the collection of objects of this type maintained by the system controller. The object itself contains the actual information about the database.

Because most systems work on a primary database, a team might decide to declare a database with the name of MAIN as its primary database. The majority of the code will use this database when requesting database operations. Then, as other data needs to be accessed, other **CmpDbsJet.Source** objects could be added to the collection for use by any code in the system. These other objects might describe databases that are in different formats or even that require different transports. The bottom line is that the application code doesn't care; it uses

the same code to accomplish tasks. In some cases, it uses the key for LOCAL instead of MAIN when it needs local data, or it uses TAXDB instead of CUSTDB, and so on.

Here is an example of a Prepare routine in which the application needs list boxes loaded with data. If a local database is present, the application would like to pull this lookup data from the local database instead of the primary server.

```
Public Sub Prepare()
On Error Resume Next
Screen.MousePointer = vbHourglass
Log.PostItem 0, "clsInterests.Prepare", STR_MSG_ENTR
SysCon.StatusSet STR_STS_PRE & gscCap_INGR
'Clear out any data currently in the combobox.
cmbTypeGrp.Clear
'If the flag for UseLocalData is set
If gicLocalDB Then
    'Use pure SQL to get data from a local database.
    Log.PostItem 0, "clsInterests.Prepare", gscLvw_INGR_Loc
    DBSvc.ExecFillLBItem DS("LOCAL"), cmbTypeGp, gscLvw_INGR_Loc
Else
    'Use a stored procedure to get data from central server.
    Log.PostItem 0, "clsInterests.Prepare", gscLvw_INGR
    DBSvc.ExecFillLBItem DS("MAIN"), cmbTypeGp, gscLvw_INGR
End If
'Set up the default.
cmbTypeGp.ListIndex = FIRST_ITEM
Log.PostItem 0, "clsInterests.Prepare", STR_MSG_EXIT
SysCon.StatusReset
Screen.MousePointer = vbDefault
End Sub
```

Notice that the code doesn't care where the data comes from. In either case, whether the data is pulled from a local database or a primary server, it will get loaded in an optimal way. This is a good example of the use of data and operator classes.

These two services are pretty skimpy. They are simply independent wrappers for the configuration settings and resource tables of an application. For the configuration service, this usually means data stored in the registry, but it might just as easily be a local database or a shared system database. The resource translation service is usually an interface to a resource-only DLL that holds the resource tables for an application. In reality, the resources might exist in multiple files or even in databases. Keeping these common interfaces ensures that the application won't be affected no matter where these pieces of data reside or in what form. Because the interface to these servers will not change by simply updating the data that is included in the file, when the servers are recompiled they can keep the same Class IDs. This means that compiled code that makes use of these servers does not need to be recompiled. You can just copy a new file to the system and register it. Any code that uses this server will never need to know that the data in the server has changed.

Like every other service, these services need to be initialized and terminated at the appropriate points. And using the interfaces should be self-explanatory. What isn't so clear is how to appropriately use these services within the context of the Layered Paradigm. That's what I'll attempt to explain in this section. I'll start at the very top with the user interface and work my way down from there.

This section includes the following topics:

There are generally two schools of thought about how to accomplish the task of separating the user interface and the data interface. Both approaches are a matter of opinion and little else. I'll illustrate and explain both, so that you can choose the one that best suits your development style. There is also an extremist perspective that I'll explain afterwards, which those of you with gobs of time and resources might actually have a use for. Unfortunately, most development shops never have enough time or resources for the extremist approach to be realistic.

The "Like a Glove" Approach

In this approach, you standardize the names of controls and forms up front. The data interface is called to populate the controls by way of a form that is either available or passed in. There are several different flavors of this approach. The following is an example of my favorite type. It is a High Schools class, and it holds all the high school records for a particular student. It shows the Initialization and the Display routines so that you can see how the caller of this function sets up the class. Previous to Display, the caller would also need to call Retrieve to make sure that the records were actually loaded into the class.

```
'This is the data interface class initialization routine.
Public Function Init(frmMeIn As Form) As Boolean
...'Set up error handling.
Set frmMe = frmMeIn
...'At this point the form is hooked up.
Init = True
End Function

'When the user of this class needs a specific record displayed
'it sets the iCurrRec property and calls the display routine.
Public Sub Display()
On Error Resume Next
Screen.MousePointer = vbHourglass
Log.PostItem 0, "clsHighSchools.Display", STR_MSG_ENTR
'Private function to clear the controls to display data in
Clear
SysCon.StatusSet STR_STS_RET & STR_OBJHSCCAP
'Do I have the record the user wants to see?
If iCurrRec >= LBound(sHsc, 2) And iCurrRec <= UBound(sHsc, 2) Then
    frmMe!txtHscCode = Trim$(sHsc(idxHSRC_ORGN_Code, iCurrRec))
    frmMe!txtHscName = Trim$(sHsc(idxHSRC_ORGN_Name, iCurrRec))
    frmMe!dtHsc.Date = Format$(sHsc(idxHSRC_LastAttendDate, iCurrRec), _
                               "Short Date")
    If CBool(sHsc(idxHSRC_Graduated, iCurrRec)) Then frmMe!chkHscGraduated
_                                                   = True
End If
Log.PostItem 0, "clsHighSchools.Display", STR_MSG_EXIT
SysCon.StatusReset
Screen.MousePointer = vbDefault
End Sub
```

Notice that the control names must be predefined. And each user interface that uses this data interface must support the same control names, must have all the controls, and so on. While this approach was the one I have used most in the past, I now do most of my work with the following approach.

The "Left Hand Doesn't Know What the Right Hand is Doing" Approach

In this approach, the separation between the layers is very distinct and intentional. The bottom line here is that at initialization time, references for the controls in the interface are passed to the data interface and the data interface just “parties” on its control references. There can be some complicated issues with this approach, but overall it's a very comfortable middle ground for developing quality, elegant solutions. It goes something like this:

```
Public Function Init(objMyParent As Object, cmbMyType As Object, _
                     cmbMyTypeGrp As Object, _
                     txtMyInt As Object, btMyIns As Object, btMyUpd As Object, _
                     btMyDel As Object, btMyClr As Object) As Boolean
On Error GoTo initacterr
Set objParent = objMyParent
Set cmbType = cmbMyType
Set cmbTypeGp = cmbMyTypeGp
Set txtInt = txtMyInt
Set btIns = btMyIns
Set btUpd = btMyUpd
Set btDel = btMyDel
Set btClr = btMyClr
Init = True
Exit Function

initacterr:
    Set objParent = Nothing
    Set cmbType = Nothing
    Set cmbTypeGp = Nothing
    Set txtInt = Nothing
    Set btIns = Nothing
    Set btUpd = Nothing
    Set btDel = Nothing
    Set btClr = Nothing
    Init = False
    Exit Function

End Function

'When the form needs to display a record
'it sets the iCurrRec property and then
'calls this routine.
Public Sub Display()
On Error Resume Next
Screen.MousePointer = vbHourglass
Log.PostItem 0, "clsInterests.Display", STR_MSG_ENTR
'Private function to clear the controls to display data in.
Clear
SysCon.StatusSet STR_STS_RET & STR_OBJINTCAP
'Do I have the record the user wants to see?
If iCurrRec >= LBound(sInt, 2) And iCurrRec <= UBound(sInt, 2) Then
    'FindItem searches a combobox for an integer and selects the item.
    FindItem cmbTypeGp, CInt(sInt(idxSINT_INGR_PKId, iCurrRec))
    FindItem cmbType, CInt(sInt(idxSINT_INCG_PKId, iCurrRec))
    txtInt = Trim$(sInt(idxSINT_Comment, iCurrRec))
End If
Log.PostItem 0, "clsInterests.Display", STR_MSG_EXIT
SysCon.StatusReset
Screen.MousePointer = vbDefault
End Sub
```

In this procedure, the private control references are used to get essentially the same work done as in the previous example. One major difference is that the control references are set only once. So if a more flexible architecture were needed, some controls could be made optional on initialization. Also, the names of the controls in the caller can be anything; they can even reside on a different form than the caller of this class.

The Extremist Approach

Just in case you have gobs of time and resources, I figured I would tell you about an ideal solution to the separation between user interface and data interface. It is a bit different than the previous two approaches. The crux of it is to have the data interface expose properties that the user interface can draw from. This complicates the user interface drastically, but it provides the biggest possible separation between the two interfaces. In this approach, there would be no initialization procedure. The user interface would call a Retrieve routine on the data interface and then start sucking the data for its controls out of the data interface. Of course, the data interface would get much smaller and dumber, but it really is just a trade-off anyway. The bottom line is that a more middle-of-the-road approach to this is where most programmers feel comfortable, but I wanted to point out that there is another alternative for those who feel so inclined.

In the previous examples, I deliberately left out how the data interface retrieves its data. This section should explain that. Essentially, the data interface should be able to think only in terms of *types* of operations it requires, and the transaction interface should handle actually *implementing* those operations. This is much like the distinction between a data class and an operator class. The transaction interface is called by the data interface with a request for data. The transaction interface, knowing the specifics of how to get the data from the external access interface, proceeds to get the data in whatever format it needs from the external access interface and feeds it to the data interface in terms the data interface can understand. In most cases, the transaction interface can just retrieve an array from the external access interface and pass this array on to the data interface. In the case of updates to data, or pulling data from multiple sources, this becomes very important.

Here is an example of the data interface creating a new record without using a transaction interface:

```
Private Sub Insert()
On Error Resume Next
Dim sParm As String

Screen.MousePointer = vbHourglass
SysCon.StatusSet STR_STS_INS & STR_OBJINTCAP
Log.PostItem 0, "clsInterests.Insert", STR_MSG_ENTR

'Build the parameter string using data from the controls.
'Interest Type
If cmbType.ListIndex = NO_ITEM Then
    MsgBox STR_MSG_REQ, vbCritical, gscAppName
    SysCon.StatusReset
    Screen.MousePointer = vbDefault
    Exit Sub
Else
    sParm = sParm & CStr(cmbType.ItemData(cmbType.ListIndex)) & STR_SPCSP
End If
'Student ID
sParm = sParm & CStr(lPKIdStu) & STR_SPCSP
'Comment
If Len(Trim$(txtInt)) Then
    sParm = sParm & sq & Left$(DoQuotes(txtInt), 255) & sq
Else
    sParm = sParm & STR_NULL
End If

Log.PostItem 0, ("clsInterests.Insert"), gscIns_SINT & sParm
'Boolean Execution Procedure; it works or it doesn't...
If Not DBSvc.ExecBool(DS("MAIN"), gscIns_SINT & sParm) Then
    SysCon.ErrSvc.Display vbCritical, STR_MSG_NOINS, STR_MSG_CNTSA
End If
Retrieve

Log.PostItem 0, "clsInterests.Insert", STR_MSG_EXIT
SysCon.StatusReset
Screen.MousePointer = vbDefault
End Sub
```

This code uses all the services well and isn't even very big. But you couldn't easily change the SQL statement that is being constructed without changing this critical code. Further, if you wanted to go to a transport that used a different flavor of SQL or didn't use SQL (such as Data Access Objects [DAO]), you would have lots of work to do. So here's what the same task would look like if you used a transaction interface:

```
Private Sub Insert()
On Error Resume Next
Dim sParm As String
```

```
Screen.MousePointer = vbHourglass
SysCon.StatusSet STR_STS_INS & STR_OBJINTCAP
Log.PostItem 0, "clsInterests.Insert", STR_MSG_ENTR

'Tell the Transaction Interface you are creating a new record.
objIntTrans.AddNew
'Set the data for the new record.
objIntTrans.Type = cmbType.ItemData(cmbType.ListIndex)
objIntTrans.StudentID = lPKIdStu
objIntTrans.Comment = txtInt
'Tell the Transaction Interface to proceed with the update.
objIntTrans.Update
'Private retrieve procedure to reload your data.
Retrieve
Log.PostItem 0, "clsInterests.Insert", STR_MSG_EXIT
SysCon.StatusReset
Screen.MousePointer = vbDefault
End Sub
```

Notice that all the smarts of building the SQL statement (or using DAO) are hidden inside the transaction interface. The data interface still manages the process, the persistence, and the flow of data to the user interface. But the work of actually performing database operations has been further isolated. Those who do lots of DAO work are probably wondering why such an abstraction layer is necessary. After all, the layer looks the same as if you had written it in DAO in the first place. But what if you needed to use stored procedures, instead of dynasets? Or what if your data was located on a mainframe? You would need to do API calls or build SQL statements to get the same tasks done. Simply using DAO would not be sufficient. By abstracting the transaction interface, you can efficiently make use of any techniques or different technologies that are required to work with your data. Regardless of whether that technique or technology is using an API, other OLE objects, or building SQL statements, this service abstracts those specifics from the rest of the application code.

If we have made a clear distinction between the data interface and the transaction interface, the role of the transaction interface should be more or less clear. And, as a side effect, the distinction between the transaction interface and external access interface should also be clearer. Essentially, the external access layer is a specific implementation of the data controller service's operator class. Back when I was talking about the data controller service, I pointed out that the operator class embodies all the specific operations that can be requested of a particular data source. In the implementation, I used a generic interface so that operator classes could be made interchangeable. This is the goal you should work toward when designing client/server components, but this goal was added to the requirements of our implementation and certainly is not dictated by the Layered Paradigm or the service model. In reality, the implementation of the transaction interface can be as simple as those sets of functions that make up your specific transaction model for accessing your data.

To emphasize even more the differences between the transaction interface and external access interface, think of the transaction interface as the piece of code that prepares the transactions and translates them from object requests into a format that the external access interface can speak. The external access interface is the physical implementation of acting out those object requests on a particular data source.

In this article, I've advanced the Layered Paradigm into the object-based arena that characterizes Visual Basic 4.0 development today. I've illustrated a model that could be used to implement the paradigm, and I've illustrated how that model could be implemented from an applications perspective. In addition, I've included the code that is an implementation of the service model explained in this article.

I want to add just a word of caution here. The code I am including is code I've used, but it is not designed for production use. It is to serve only as a model for you in developing your own solutions. Of course, by creating your own implementation, you will invariably come to understand this at a much deeper level as well. Which is, by the way, a good thing.

Anyway, you now have all you need to start leveraging the Layered Paradigm and the service model.

The system controller service wraps all the previous services up in a nice nutshell for the application. The application can use any of the separate services independently if it likes, but then it will need to understand the dependencies between them. So the primary focus of the system controller service is to wrap these servers all together. Then the application needs to worry only about getting an instance of the system controller service, and it automatically gets all the other services.

The secondary focus of the system controller service is to encapsulate any code that an enterprise wants to share at the system level. For example, the system controller might encapsulate status displays, initialization and version control procedures, or notification systems — anything that is used consistently by multiple users of the system controller, even if they are not necessarily within the same application. In the sample associated with this article, I have made the system controller responsible for handling the status and percent displays of the application. So any service or code in the application that has access to the system controller service can call the status and percent routines and know that the status and percent displays will handle the requests in an appropriate manner.

That pretty much wraps up my discussion of the services. Let's move on to how an application can use these services to actually get some work done.

The Internet Transfer control automatically sets itself to the correct protocol, as determined by the *protocol* portion of the URL. Therefore, when you use the **OpenURL** or **Execute** method, you don't need set the **Protocol** property.

If a component is moved to a new file location, or its CLSID changes, Microsoft Transaction Server must be updated with the new information. An easy way to ensure that Transaction Server is using the latest information is to select the **Refresh All Components** command from the **Tools** menu.

If you need to force package processes to shut down, you can do so by right-clicking **My Computer** and choosing **Shutdown Server Processes**.

The **Request** object retrieves values from the user's browser, and passes the information to the Web server in an HTTP request message.

For more information, see Intrinsic Objects.

The **Response** object controls what information is sent to a user in the HTTP response message.

For more information, see Intrinsic Objects.

The **Session** object stores information about a particular user session.

For more information, see Intrinsic Objects.

The **Server** object provides access to resources that run on a Web server.

For more information, see Intrinsic Objects.

A **Session** object is created when a user requests an Active Server Page from the Web application.

For more information, see The Session Object.

A **Session** object is created when a user requests an Active Server Page from the Web application.

For more information, see The Session Object.

A **Session** object is created when a user requests an Active Server Page from the Web application.

For more information, see The Session Object.

A **Session** object is created when a user requests an Active Server Page from the Web application.

For more information, see The Session Object.

The Session_OnStart event procedure runs when a user requests an Active Server Page from your Web application.

For more information, see Using Events in the Global.asa File.

The Session_OnEnd procedure will run only if there is an active session when you shut down the WWW service of your Web server.

For more information, see Using Events in the Global.asa File.

The Application_OnStart event procedure runs when you start the WWW service of your Web server.

For more information, see Using Events in the Global.asa File.

The Application_OnEnd event procedure runs when you shut down the WWW service of your Web server.

For more information, see Using Events in the Global.asa File.

The **Form** collection retrieves the values of form elements that have been sent by a form that uses the POST method.

For more information, see Using the Form Collection.

When you do not refer to a specific collection of the **Request** object, the Web server searches the collections. The server starts searching the **QueryString** collection, and then searches the **Form** collection.

For more information, see The Request Object.

The **QueryString** collection retrieves the values of variables in the HTTP request message that have been sent by a form that uses the GET method.

For more information, see Using the QueryString Collection.

The **ServerVariables** collection retrieves the values of environment variables.

For more information, see Using the Form Collection.

Cookies are sent by the Web browser as part of an HTTP request; session variables reside on the Web server.

For more information, see Using Cookies.

Cookies are saved in a file on the user's computer; session variables are saved on the Web server.

For more information, see Using Cookies.

You can create both cookies and session variables with server-side script.

For more information, see Using Cookies.

You can specify an expiration time for a cookie. The browser will destroy it when the time expires. Session variables are destroyed when the session ends.

For more information, see Using Cookies.

You cannot use the **New** keyword in server-side script.

For more information, see The Server Object.

Installing an ActiveX server component is the first step in accessing properties and methods. However, you cannot use the **New** keyword in server-side script.

For more information, see The Server Object.

Install the ActiveX server component, and use the **CreateObject** method of the **Server** object to instantiate an ActiveX server component.

For more information, see The Server Object.

You cannot install an ActiveX server component on the user's computer.

For more information, see The Server Object.

The <% %> tag is used for a statement that will run on the Web server.

For more information, see Creating Active Server Pages in Chapter 2: Developing a Web Project.

The <SCRIPT> tag creates client-side script, unless you set the RUNAT attribute to Server.

For more information, see Creating Active Server Pages in Chapter 2: Developing a Web Project.

To set the language for server-side script, you must use the <%@ %> tag.

For more information, see Creating Active Server Pages in Chapter 2: Developing a Web Project.

To set the language for server-side script, you must use the <%@ %> tag.

For more information, see Creating Active Server Pages in Chapter 2: Developing a Web Project.

The users will need valid NT accounts.

For more information, see Preventing Anonymous Logon.

If you turn off Anonymous logon, only users with valid NT accounts can access your Web site.

For more information, see Preventing Anonymous Logon.

If users know the password for the IUSR_*computername* account, they can view your Web site because it is a valid NT account. However, the password is generated by IIS and cannot be found easily.

For more information, see Preventing Anonymous Logon.

Any user who logs on with a valid NT account will be able to see the pages of your Web site.

For more information, see Preventing Anonymous Logon.

To limit database access to users of your Web site, create an ID that has limited rights to your database, and use that logon ID to access the database.

For more information, see Controlling Access to a Database.

Records in an SQL database do not have any effect on who has access to NT or SQL resources.

For more information, see Controlling Access to a Database.

Hard coding an account and password is dangerous because it exposes the unrestricted administrative account for the SQL Server.

For more information, see Controlling Access to a Database.

Disabling these accounts guarantees only that users with valid NT accounts can access the NT computer. It does not change the database access rights.

For more information, see Controlling Access to a Database.

The Web browser tries Anonymous authentication, but several other types of authentication are also tried in order to gain access.

For more information, see Preventing Anonymous Logon.

Anonymous authentication is tried first.

For more information, see Preventing Anonymous Logon.

The Web browser tries Anonymous, NT Challenge/Response, and then Basic authentication.

For more information, see Preventing Anonymous Logon.

If you have enabled all three authentication methods, the Web browser tries Anonymous authentication when a user requests a page. If that fails, the user must provide a valid logon ID and password. If the browser supports NT Challenge/Response authentication, the Web browser tries to use it. Otherwise, it tries to use Basic authentication.

For more information, see Preventing Anonymous Logon.

To require digital certificates from users, configure the virtual directory of your Web site to use the Secure Sockets Layer (SSL).

For more information, see Requiring Certificates.

Secure Sockets Layer (SSL) is a virtual directory setting, not a setting for the entire Web server.

For more information, see Requiring Certificates.

Anonymous logons provide user names and passwords for the HTTP request. Digital certificates are exchanged as part of the Secure Sockets Layer (SSL) protocol.

For more information, see Requiring Certificates.

The **Session** object does not authenticate users, and therefore, it will not use digital certificates.

For more information, see Requiring Certificates.

Only an NTFS partition has the required security settings to enable this type of restriction.

For more information, see Setting NTFS Permissions.

To allow all users to access your Web site, enable Anonymous logon. To restrict access to the file private.asp, remove the Anonymous logon account from the access control list for the file private.asp.

For more information, see Setting NTFS Permissions.

To allow all users to access your Web site, enable Anonymous logon, not the Guest account.

For more information, see Setting NTFS Permissions.

If you set the HIDDEN attribute for a file, users can still access the file. To control access to a file, set NTFS file permissions for the file.

For more information, see Setting NTFS Permissions.

There is no **Server** collection of the **Request** object. To retrieve the name of the logon account, use the **ServerVariables** collection of the **Request** object.

For more information, see Preventing Anonymous Logon.

To retrieve the name of the logon account, use the **ServerVariables** collection of the **Request** object, and not the **ClientCertificates** collection.

For more information, see Preventing Anonymous Logon.

To retrieve the name of the logon account, use the **ServerVariables** collection of the **Request** object.

For more information, see Preventing Anonymous Logon.

The logon account is not stored in the **Session** object.

For more information, see Preventing Anonymous Logon.

The ACID test is a set of properties that a transaction should have for it to succeed.

For more information about the ACID test, see Transaction Processing Concepts.

Change of state refers to a transaction. A transaction changes a set of data from one state to another.

The ACID test is a set of properties that a transaction should have for it to succeed.

For more information about the ACID test, see Transaction Processing Concepts.

The ACID test is a set of properties that a transaction should have for it to succeed.

For more information about the ACID test, see Transaction Processing Concepts.

The properties are atomicity, consistency, isolation and durability.

For more information about the ACID test, see Transaction Processing Concepts.

You set the transaction property of the component, not the package.

For more information, see Adding Components to a Package.

Setting the transaction property to "Supports Transactions" indicates only that a Microsoft Transaction Server component can participate in a transaction, but does not require that it does.

For more information, see Adding Components to a Package.

To set a transaction component to require a transaction, select the **Requires a Transaction** option on the **Transactions** tab of the **Properties** dialog box.

For more information, see Adding Components to a Package.

When an object is created, Microsoft Transaction Server creates a corresponding **Context** object. A **Context** object eliminates the need to call any transaction functions, such as **BeginTrans** or **EndTrans**.

For more information, see Adding Components to a Package.

A **Context** object eliminates the need to call any transaction functions, such as **BeginTrans** or **EndTrans**.

Instead, each public method of your business object calls **SetComplete** or **SetAbort** to indicate success or failure.

For more information, see Adding Transactional Support.

Each public method of your business object calls the **SetComplete** or **SetAbort** method of the **Context** object to indicate success or failure.

For more information, see Adding Transactional Support.

Placing a component in the Microsoft Transaction Server Explorer will register it with Microsoft Transaction Server, but will not modify the component to support transactions.

For more information, see Adding Transactional Support.

The methods **SetComplete** and **SetAbort** are implemented by the **Context** object, so you do not need to implement them for the Transaction Server component.

For more information, see Adding Transactional Support.

This answer is partially correct. The advantages of stateless objects include all of the answers to the question.

For more information, see Creating a Stateless Object.

This answer is partially correct. The advantages of stateless objects include all of the answers to the question.

For more information, see Creating a Stateless Object.

This answer is partially correct. The advantages of stateless objects include all of the answers to the question.

For more information, see Creating a Stateless Object.

The advantages of stateless objects include all of the answers to the question.

For more information, see Creating a Stateless Object.

The **GetObjectContext** API function returns a reference to the **Context** object.

For more information, see Adding Transactional Support.

You cannot use the **CreateObject** function to retrieve a reference to the **Context** object.

The **Context** object is created by Microsoft Transaction Server. You use the **GetObjectContext** API function to set a reference to it.

For more information, see Adding Transactional Support.

Microsoft Transaction Server creates a **Context** object but does not pass a reference to it back to your application.

For more information, see Adding Transactional Support.

You cannot use the **GetObject** function to retrieve a reference to a **Context** object created by Microsoft Transaction Server.

For more information, see Adding Transactional Support.

ActiveX controls are in-process, but they have a graphical interface and other features that are not necessary for an ActiveX server component.

For more information, see Choosing the Type of Component.

ActiveX DLLs are in-process, and can be used as ActiveX server components.

For more information, see Choosing the Type of Component.

The ActiveX EXE Visual Basic project template is an out-of-process ActiveX server component.

For more information, see Choosing the Type of Component.

A Visual Basic Standard EXE project template is a standard Windows application. It is not an ActiveX server component.

For more information, see Choosing the Type of Component.

Building an ActiveX server component isolates the process from changes to user and data services.

For more information, see Business Objects.

This strategy will work only with Web browsers as clients. A better solution will work with multiple kinds of clients.

For more information, see Business Objects.

Although this strategy will work, it is isolated to a specific type of database. If the business process is modified to include multiple databases, the process will need to be updated. If the single database is changed to a different type, it also requires the process to be updated.

For more information, see Business Objects.

Although this strategy will work, it places all processes in a single .exe file. The file cannot be distributed across multiple computers, and does not scale well.

For more information, see Business Objects.

In Visual Basic, unattended execution indicates that the project will not interact with the user, and makes the application apartment threaded.

For more information, see Creating Methods for Classes.

The **Instancing** property determines if one or multiple clients can use the same instance of a component.

For more information, see Creating Methods for Classes.

The **Public** keyword causes a method to be available outside of the component.

For more information, see Creating Methods for Classes.

Setting the project type does not enable methods to be exported.

The project type specifies whether the project is an ActiveX DLL, ActiveX EXE, or standard Windows program.

For more information, see Creating Methods for Classes.

You can register an in-process component by compiling it in Visual Basic, running RegSvr32.exe, or by running a Setup program for the component.

For more information, see Registering a Component.

You can register an in-process component by compiling it in Visual Basic, or by running a Setup program for it. However, you can also use other registration methods.

For more information, see Registering a Component.

You can register an in-process component by compiling it in Visual Basic or by running RegSvr32.exe. You can also use other registration methods.

For more information, see Registering a Component.

You can register an in-process component by running a Setup program for it or by running RegSvr32.exe. You can also use other registration methods.

For more information, see Registering a Component.

A business rule provides guidelines for how business activities should occur, and for how to ensure their integrity.

For more information, see Business Rules and Business Processes.

A business process is a sequence of related tasks that produce a specific response to a user's request.

For more information, see Business Rules and Business Processes.

Formally known as Automation servers, ActiveX server components provide component functionality, such as a business rule.

For more information, see Business Rules and Business Processes.

A business object implements business processes and business rules.

For more information, see Business Rules and Business Processes.

The Data Form Wizard generates .asp files, not .htm files.

For more information, see Viewing Data Form Files.

```
Set EnrollObj = Server.CreateObject("StateU.Enrollment")
errEnrollment = EnrollObj.Add(lngStudentID,strClassID)
```

```
Public Function Drop(ByVal lngStudentID As Long, _
    ByVal strClassID As String) As Integer

    On Error GoTo ErrorHandler
    Dim conn As ADODB.Connection
    Dim strSQL As String

    Drop = 0
    Set conn = CreateObject("ADODB.connection")
    conn.Open "DSN=StateU;UID=sa;PWD=;"

    'Make sure class isn't already completed
    If ClassCompleted(conn, lngStudentID, strClassID) Then
        Drop = 1
        Exit Function
    End If

    'Perform the work of dropping the student from the class
    strSQL = "DELETE FROM enrollment where ClassID ='" & _
        strClassID & "' AND Studentid =" & lngStudentID
    conn.BeginTrans
    conn.Execute strSQL
    conn.CommitTrans
    conn.Close
Exit Function

ErrorHandler:
    If Not IsEmpty(conn) Then conn.RollbackTrans
    Drop = 1
End Function
```

```
Public Function Transfer(ByVal lngStudentID As Long, _
    ByVal strSrcClassID As String, _
    ByVal strDstClassID As String) As Integer

    On Error GoTo ErrorHandler
    Dim conn As ADODB.Connection
    Dim strDropSQL, strAddSQL As String

    Transfer = 0
    Set conn = CreateObject("ADODB.connection")
    conn.Open "DSN=StateU;UID=sa;PWD=;"

    'Create update strings
    strDropSQL = "DELETE FROM enrollment where classid = '" _
        & strSrcClassID & "' and studentid = " & lngStudentID
    strAddSQL = "INSERT INTO enrollment (ClassID, StudentID)" & _
        " VALUES ('" & strDstClassID & "'," & lngStudentID & ")"

    If ClassCompleted(conn, lngStudentID, strSrcClassID) Then
        Transfer = 1
        Exit Function
    End If

    'Perform work of transferring student
    conn.BeginTrans
    conn.Execute strDropSQL
    conn.Execute strAddSQL
    conn.CommitTrans
    conn.Close
Exit Function

ErrorHandler:
    If Not IsEmpty(conn) Then conn.RollbackTrans
    Transfer = 1
End Function
```

Click here to connect to the Microsoft Visual InterDev Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Use all of the new content packed into this Web site and become a Visual InterDev expert. You'll find white papers, sample applications, demos, a troubleshooting guide, new training information, and more.

Click here to connect to the Microsoft FrontPage 97 Web site.
{ewc mvimg, mvimage,!intjump.bmp}

Microsoft FrontPage makes creating professional-quality Web sites effortless with powerful new functionality, support for the latest Web technologies, and great integration with Microsoft Office. It's never been easier to develop and maintain your own professional Web sites! Visit this Web site to find out more.

By Delane Hewett

What is it that separates an aesthetically pleasing Web site from a not-so pleasing site? More often than not, a great site is identified by its distinctive and consistent appearance. Many such Web sites are developed by teams or individuals with skills in both Web technologies and the graphic arts. Unfortunately, many of us don't have the luxury of delegating for skills that we don't possess ourselves. Some of us who can throw together a Web site that is pretty impressive from a technical perspective are often disappointed by our inability to get the visually professional look that today's sites demand.

Visual InterDev themes can help you achieve that consistent, professional look that so often eludes the graphically-impaired. (The more talented can use them just to make things easier.) Themes combine a recent W3C Web technology called Cascading Style Sheets (CSS) with graphical elements such as background images, headings, rules, bullets, and buttons to give your Web site a visual flair that will attract users. For more information about Cascading Style Sheets, see the W3C Working draft "Cascading Style Sheets, level 1" at http://www.w3.org/pub/WWW/TR/WD-css1.html.

This white paper includes the following topics:

Delane Hewett is a Program Manager on Visual InterDev. He is responsible for extensibility, wizards, Design-Time Controls, and other upcoming features. Previously, he wrote the Internet section of *Tricks of the Visual Basic Gurus* by Sams Publishing, 1996.

A theme, in the Visual InterDev sense, is a collection of graphic images, formatting, fonts, and colors that look good together. When you apply a theme to your Web site, these elements are applied to all the pages in your site, making them all look good together. A set of pre-designed themes comes with Visual InterDev. You can also install third-party themes or create your own.

Theme elements include:

® Fonts (managed through Cascading Style Sheets)

® Background images

® Heading images

® Rule images

® Bullet images

® Navigational images, such as buttons

These elements are stored in a directory structure which defines the theme. You'll find an example of the directory structure of one of Visual InterDev's pre-designed themes, Redside, below.

```
Themes
    Redside
        Preview.bmp
        StyleSheets
            Style1.css
            Style2.css
        Backgrounds
            Back1.jpg
            Back2.jpg
            Back3.jpg
        Headings
            Head1.jpg
            Head2.jpg
            Head3.jpg
        Bullets
            Bullet1.gif
            Bullet2.gif
            Bullet3.gif
        Navigation
            Button1.gif
            Button2.gif
            Up.gif
            Down.gif
            Right.gif
            Left.gif
            Nav1.gif
            Nav2.gif
            Nav3.gif
        Rules
            Rule1.gif
            Rule2.gif
            Rule3.gif
```

The name of the parent directory is the name of the theme. The directory structure and the file names remain the same for all themes. Only the theme name (that is, parent directory name) is different.

Creating and Modifying Themes

To create a theme, all you have to do is create a directory with your theme's name and create at least one subdirectory with the same structure and name as one of those in the example above. When you add your

graphical elements and style sheets, give them the exact names called for in the structure (that is, Style1.css, and so on), and place the files in the appropriate directories.

You can create your own style sheets (See Appendix A), modify the graphic images, or use Microsoft Image Composer to create new graphical elements for your own theme. Just add them to the appropriate directories and you're on your way.

You can also customize one of the pre-designed Visual InterDev themes or install third-party themes into your Themes directory. To modify any one theme, you simply go into the directories that contain the images that you want to modify. Replace the old files with your new ones (keeping the same names). If you don't want to use all the possible elements in a theme, just delete the existing files from the directory — Visual InterDev ignores missing files and directories — and, voilá, you've got a customized theme.

Theme Location, Where Do These Things Live?

If you use the Data Form wizard or certain templates to create a Web page, you will be prompted to choose a theme from a theme list. (See "Creating Your Own Visual InterDev Templates" for more information.) If your custom theme directory structure is under the default Themes directory, your theme will appear in this theme list.

If you chose the Typical Setup option during installation, themes are stored in a directory called "Themes" under the install directory for Visual InterDev. To change the default directory:

1. Open the registry to the following node: [HKEY_CURRENT_USER\\Software\Microsoft\DevStudio\5.0\ Directories]
2. Change the value of ThemeDir to the new default directory.

For example, the following setting indicates a new default directory of \\Server\Share\MyThemes:

[HKEY_CURRENT_USER\\Software\Microsoft\DevStudio\5.0\Directories]

"ThemeDir" = "\\Server\Share\MyThemes"

There are various ways that you can use themes and their components in your Web site.

® You can incorporate themes into your Web site through the Visual InterDev templates.

® You can use individual components of themes in a specific Web page.

® You can use themes with Active Server Pages to increase your options in dynamic Web sites.

Using Themes with Visual InterDev Templates

The Template Page wizard in Visual InterDev makes using themes a very easy process. If you use a template page which contains the string "<%#ThemeName#%>", the wizard will prompt you to choose one of the themes from your default Themes directory.

For example, you could add the following line to a template in the default Templates directory: <SRC=<%#THEMENAME#%>/Bullets/Bullet2.jpg>

When the wizard processes the template, you are prompted to choose a theme. If you choose, for example, your High Tech Corporate theme, the following line appears in your HTML text:

<SRC= High%20Tech%20Corporate/Bullets/Bullet2.jpg>

Note HTTP paths should not have blanks. Substitute “%20” for the blank space.

Using Theme Components Directly

Theme components are stored on your file system and therefore can be used in any of your Web pages. You don’t need to do anything special, just add them to your project. Each image and style sheet is stored in a directory named after its corresponding theme.

Using Themes with Active Server Pages

If you have a dynamic site composed of Active Server Pages, you might like to have all your pages share a theme, but want to maintain the ability to switch between themes.

Consider building your site using themes stored to session properties throughout all the pages. You can set the session property equal to the theme’s name. This way, the actual name of the theme occurs in only one place in your Web site, but every page shares that theme through the session. The following is an example of a simple page using a theme based on a session property. The bold text is the code representing the theme.

```
<%
If Session("Theme") is Nothing then
Theme = "defaulttheme"
Else
Theme = Session("Theme")
End If
%>

<HTML>
    <HEAD>
        <LINK REL=STYLESHEET HREF="./stylesheets/<%=Theme%>/style1.css">
        <TITLE>Here is the Title </TITLE>
    </HEAD>
    <BODY BACKGROUND="./images/<%=Theme%>/Background/back1.jpg"
alink="#FF000" BGCOLOR="#FFFFFF" TOPMARGIN=20 LEFTMARGIN=0>
        <Center>
            <H1>Hello World! </H1><BR>
            <P><IMG SRC="./images/<%=Theme%>/Rules/Rule1.gif"
ALT="[Horizontal Rule Image]" ALIGN="Bottom"
WIDTH=80% HEIGHT="25"></P>
```

```
        </Center>
    </BODY>
</HTML>
```

```
Your Global.asa file would contain an entry such as:
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Session_OnStart
    Session("Theme") = "RedSide"
End Sub
</SCRIPT>
```

Themes won't solve all the problems of a graphically-impaired Web site developer. They won't help you with aspects of page layout such as frames, nor will they help you incorporate multimedia components. But using themes, you can bring a professional, distinct, consistent look to your Web sites.

Below is an example of a simple linked style sheet used by a Visual InterDev theme. This example contains the minimum styles which need to be defined for a theme-driven page design. Thus, it can be used as a template for your own style sheets.

Remove the comment lines before you use the style sheet. There are some attributes of the W3C style sheet spec that are not yet supported by Microsoft Internet Explorer 3.0, notably: letter-spacing and some of the more interesting flexible background image tiling properties.

Visual InterDev themes contain two style sheets. In Style1.css we define the <p> tag as 12pt/15pt "Arial", in Style2.css we have 12pt "Arial" (with no leading defined).

Caution Style2.css breaks any page that has a <form> tag on it. (It makes it look like the form is off the page.)

```
/* set the left margin for rules (lines) and text not enclosed in <p> tags */
BODY {
    margin-left: 0px;
    text-align: left;
    background: transparent;
    border: black;
    color: gray
    }

/* set the left margin and attributes for general text */
 P {
    color: darkGray;
    text-indent: 0in;
    font: 8.5pt/23pt "Verdana"
    line-height: -3in;
    margin-left: 1.5in;
    }

 LI {
    display: list-item;
    margin-left: .5in
    }

 DL {
    display: list-item;
    margin-left: 2.5in
    }

 DT {
    display: list-item;
    text-align: left;
    margin-left: 1.5in;
    font: 9pt/10pt "Verdana";
    font-weight: bold;
    color: navy
    }

 DD {
    display: list-item;
    font: 9/15pt "Verdana";
    text-align: left;
    margin-left: 1.5in;
    color: gray
    }
```

```
 H1 {
    font: 22pt/20pt "Verdana";
    background: transparent ;
    margin-left: .25in;
    color: navy
    }

 H2 {
    font: 16pt/18pt "Verdana";
    background: transparent ;
    margin-left: .25in;
    color: navy
    }

 H3 {
    font: 14pt/14pt "Verdana";
    background: transparent ;
    margin-left: .25in;
    font-weight: bold;
    color: navy
    }

 H4 {
    font: 12pt/12pt "Arial";
    background: transparent;
    margin-left: 1.5in;
    color: navy
    }

 H5 {
    font: 9pt/10pt "Arial";
    margin-left: 1.5in;
    font-weight: bold;
    background: transparent;
    color: gray
    }


 B, STRONG {
    font-weight: bold
    }
 I, CITE, EM, VAR, ADDRESS, BLOCKQUOTE {
    font-style: italic
    }

 PRE, TT, CODE, KBD, SAMP {
    font-family: monospace
    }

/* the link colors will be overwritten for the nav bars (they are hardcoded to
white) */
 A:link {
    color: coral;
    text-decoration: none
    }

 A:visited {
```

```
   color: red ;
   text-decoration: none
   }

A:active {
   color: orange ;
   text-decoration: none
   }
```

Below is the list of necessary elements for Visual InterDev themes. Because the names of the elements are hardcoded into the wizard-generated HTML pages/templates, the directory structure is an important part of making the theme work effectively.

Element	File Name	Size	Positioning	Notes
Style sheet	Style1.css	N/A	N/A	Controls alignment, font and font attributes for HTML tags.
				Font list should be limited to those that either ship with Windows 95 or with IE3.0: Arial, Arial Black, Times New Roman, Courier New, Verdana, Impact. (See http://www.microsoft.com/truetype/ieplor/ie3b2.htm for more information on fonts.)
				Note: A left-margin default is hardcoded into the HTML page. If the pages are loaded into a style browser, they will still be useable.
	Style2.css			Same as Style1.css but focused on tabular output. The same as Style1 without the leading.
Background	Back1.jpg	Varied	IE3 doesn't yet support calling this from the style sheet	These are .jpg files, which ensure a smoother appearance for background images.
				Note: The .jpg files should be tested in a 256-color system to ensure that they look acceptable when tiling (that they dither seamlessly).
	Back2.jpg			Same as Back1.jpg but left side image effects are not allowed.
Graphic header	Header1.gif	Varied	Left margin as defined by <body> margin	Site wizards will include a link to Header1.gif at the beginning of default.htm (home page). There are no height= and width= tags in the html, so the image sizes can be flexible, but will therefore load more slowly.
				The top-margin of the html page is =0, so the header will butt up against the top of the browser window. Theme designers can build in a transparent background at top of image if space is needed.
				Note: Themes with decorative side margins may not need graphic headers. Because the Header1.gif is required, theme designers should substitute a transparent spacer .gif if they want no image to show.
	Header2.gif			Same as Header1.gif. This is an optional element.
Navigation	Nav1.jpg	Varied	Left margin as defined by <p> margin	Image tiles horizontally behind navigation table row (entire row; not cell-by-cell).
				Note: Links in navigation tables are hardcoded white, so Nav1.jpg should have enough contrast for easy readability.
	Button1.gif			Large blank button that you may place text on.
	Button2.gif			Smaller blank button that you may place text on.
	Down.gif			Localization-safe Down button.
	Left.gif			Localization-safe Left button.
	Right.gif			Localization-safe Right button.
	Up.gif			Localization-safe Up button.
Horizontal rules	Rule1.gif	650x25	Left margin as defined by <body> margin	Sizes are hardcoded into the html pages. Varied sizes are derived from using gif 89a format with transparent margins.
	Rule2.gif	500x15		
Bullets	Bullet1.gif	20x20	Left margin as defined by <dl>,	As with rules, apparent varied sizes are derived by using gif 89a format with transparent margins.
	Bullet2.gif	15x15		

	Bullet3.gif	15x15	<dt>,and <dd> margins	
Preview bitmap	Preview.bmp	142x246	N/A	Colors: 256 Windows-system palette. Preview bitmaps should represent the style as well as possible. Preview images may or may not be localized. If you don't want to localize them, don't include localization text. Preview bitmaps should have a sunken border treatment at edge of image. Use a template or existing bitmap to ensure border consistency between previews (the control on the wizard screen is flat to accommodate static art on other pages).

By Delane Hewett

Microsoft Visual InterDev templates give you models for creating various sorts of Web pages. Templates are Hypertext Markup Language (HTML) or Active Server Page (ASP) files constructed so that you can put specific information into a predetermined structure.

A Visual InterDev template can include HTML layouts, HTML text, and server and/or client side scripting. In addition, you can customize the template by imbedding placeholders (called Replaceable Parameters or RPs) in it. When you run the Template wizard, these RPs will be replaced with HTML and/or scripting.

A template can be a single file or a group of up to three associated files. These files can be HTML, an ASP, and/or an ALX (Active Layout). If multiple files are used for a template, they must all have the same name, but will have different file extensions (for example, htm, asp, alx). For example, the following files would belong to a single template named “Home”: Home.htm, Home.asp, Home.alx.

The wizard will display the name only once in the Templates list, but will act on each of the files with the same name. Only one will be opened in the editor at the completion of the wizard, based on the following table.

Combination	Open this file
ASP, ALX, HTM	ALX
ASP, HTM	HTM
HTM, ALX	ALX
ASP, ALX	ALX
HTM	HTM
ASP	ASP
ALX	ALX

Note The Template Wizard assumes that the HTML or ASP files have the correct object tags in them pointing to the ALX file.

All will be parsed for Replaceable Parameters and added to the project.

Why Use a Multiple-File Template

There are two primary scenarios for multiple-file templates. In the first scenario, you combine an HTM file with an Active Layout (ALX) file. In order to use active layout files in web sites, you need an HTM file to store the W3C “Object” tag (pointing to the ALX file) and you need an ALX file to store the ActiveX controls and their 2-D placement. Thus, you need two files, HTM and ALX, for your template.

In the second scenario, you have an HTM file containing an HTML form that you would like processed by an ASP file. Here you would also need two files, this time an HTM and an ASP file.

Using Replaceable Parameters (RPs)

An RP is a placeholder in the template that the Template wizard recognizes as something it needs to replace. As the Template wizard runs, it identifies each RP and, depending on the RP, either prompts the user or takes actions itself. An RP is surrounded by the following delimiters “<%#“ and “#%>”. For example, a replaceable parameter might look like this:

```
<%#This is a Replaceable Parameter#%>
```

Which RPs Are Recognized by the Template Wizard?

RPs are case insensitive. The following RPs are automatically recognized by the Template wizard.

1. <%#THEMENAME#%>
2. <%#DATACONNECTION#%>
3. <%#FILENAMEWITHOUTEXTENSION#%>

4. <%#FILENAMEWITHEXTENSION#%>

Using RPs for Themes

The Replaceable Parameter <%#THEMENAME#%> can be placed anywhere in an HTM or ASP template file. When the Template wizard encounters this RP, it prompts you to select a theme that will be applied to your template. For more information about themes, see “Creating Your Own Visual InterDev Themes.” The following represents an HTML document fragment before and after processing by the wizard. In this example, the user chose a theme named “swamp."

Before: <A IMG=”./images/<%#THEMENAME#%>/rules/rule1.gif”></A>

After: <A IMG=”./images/swamp/rules/rule1.gif”></A>

No matter how many times the ThemeName Replaceable Parameter is found within the template, you will be prompted to choose a theme only once.

Using RPs for Data Connections

The Replaceable Parameter <%#DATACONNECTION#%> can be placed in any ASP file. This RP is used by the wizard to allow selection of an existing project Data Connection. The user is prompted with a list of existing Data Connections to choose from.

Using RPs for File Names

The Replaceable Parameters <%#FILENAMEWITHEXTENSION#%> and <%#FILENAMEWITHOUTEXTENSION#%> are used to insert the template’s file name into the document created by the template. You can specify the file name with or without the file extension.

Defining Your Own Replaceable Parameters

If an RP is not known to the Template wizard, the wizard assumes that it is a user-defined RP and prompts you for the substitution string. The following example illustrates the use of a user-defined RP intended to collect the user’s Email alias:

```
<H5>Send mail to <A HREF=“MAILTO: <%#Enter your Email alias here:#%>”> <%# Enter
your Email alias here:#%></A></H5>
```

If the user types “johndoe@mycompany.com” in the prompt, the file would contain the following text:

```
<H5>Send mail to <A HREF=”MAILTO: johndoe@mycompany.com”> johndoe@mycompany.com
</A></H5>
```

Note The maximum text length is 100 characters.

Template Locations

If you chose the Typical Setup option during installation, templates are stored in a directory called “Templates” under the install directory for Visual InterDev. To change the default directory:

® Open the registry to the following node:

[HKEY_CURRENT_USER\\Software\Microsoft\DevStudio\5.0\Directories]

® Change the value of TemplateDir to the new default directory.

For example, the following setting indicates a new default directory of \\Server\Share\MyTemplates:

```
[HKEY_CURRENT_USER\\Software\Microsoft\DevStudio\5.0\Directories]
    “TemplateDir” = “\\Server\Share\MyTemplates”
```

Static Files Referenced in Templates

Static files, such as in-line images, referenced in a template through relative URLs (for example, /images/arrows/arrow1.gif) will be imported into the project maintaining the path to their relative locations. If the static file is not contained in a directory under the template in the file system, then the file will be copied to the Images directory of the project. The link in the template file will be modified to reference the static file in its new relative position. Static files referenced through absolute URLs (for example, http://myserver/myshare/images/arrow1.gif) are not added to the project.

A single ALX file may be a template but usually requires an associated HTM file.

Delane Hewett is a Program Manager on Visual InterDev. He is responsible for extensibility, wizards, Design-Time Controls, and other upcoming features. Previously, he wrote the Internet section of *Tricks of the Visual Basic Gurus* by Sams Publishing, 1996.

Click here to launch the HTML Reference Library.
{ewc mvimg, mvimage,!exec.bmp}

The HTML Reference Library, provided by Stephen Le Hunte, is the most definitive reference for currently useable HTML elements. It directly supports Internet Explorer, Netscape, and Mosaic.

System Requirements

The HTMLib requires Internet Explorer 3.01 (or above) with the HtmlHelp ActiveX controls installed for it's basic functionality. Other ActiveX controls are employed throughout the HTMLib (for examples), but these are controls that should be installed with the Internet Explorer package (the label, timer, HTML Layout controls). The HtmlHelp controls are available from http://www.microsoft.com/workshop/author/htmlhelp/. Click here to connect to the HTMLHelp Web page.
{ewc mvimg, mvimage,!intjump.bmp}

Note This program is in the \Tools\HTMLib directory on the *Mastering Web Site Development CD-ROM*.

Stephen Le Hunte is an independant software developer specializing in the development of on-line user assistance and information systems using Microsoft products and technologies. In his spare time, he's trying to finish a PhD research degree. Stephen can be reached at cmlehunt@swan.ac.uk or via the URL http://hot.virtual-pc.com/htmlib/.

Use the MakeCert program to generate a test X.509 certificate. The program does the following:

1. It creates a public/private key pair for digital signatures and associates it with a name that you choose.
2. It associates the key pair with a publisher's name that you choose.
3. It creates an X.509 certificate, signed by the root key or one you specifiy, that binds your name to the public part of the key pair. If you do not specify a root key, MakeCert generates one for you.

The syntax for invoking MakeCert is:

MAKECERT [*options*] *outputfile*

where the options are:

- **-u:***subjectKey* is the location of the publisher's key pair name. If a key pair does not exist, one is created.
- **-k:***subjectKeyFile* is the location of the publisher's .pvk file.
- **-n:***name* is the name for the publisher's certificate. This name must conform to the X.500 standard. The easiest way to do this is to use the form "CN=*MyName*."
- **-d:***displayname* is the display name of the publisher in the SPC UI.
- **-s:***issuerKeyFile* is the location of the issuer's key. The default is the test root key.
- **-i:***issuerCertFile* is the location of the issuer's certificate.
- **-l:***policyLink* is a link to SPC agency policy information (for example, a URL).
- **-U:***subjectCertFile* is the certificate file name with the existing subject public key to be used.
- **-#:***serialNumber* is the serial number of the certificate. The maximum value is 2^31. The default is a value generated by the program that is guaranteed to be unique.
- **-I** means the certificate will be used by individual software publishers.
- **-C** means the certificate will be used by commercial software publishers.
- **-C:f** means the certificate will be used by commercial software publishers who have met the minimum financial criteria.
- **-x:***providerName* is the CryptoAPI provider to use. (See the CryptoAPI documentation for more details.)
- **-y:***nProviderType* is the CryptoAPI provider type to use. (See the CryptoAPI documentation for more details.)
- **-K:***keyspec* is the key specification. This parameter can either be S for a signature key (this is the default) or E for a key-exchange key.
- **-B:***dateStart* is the date when the certificate first becomes valid. The default is when the certificate is created.
- **-D:***nMonths* is the duration of the validity period.
- **-E:***dateEnd* is the date when the validity period ends. The default is the year 2039.
- **-h:***numChildren* is the maximum height of the tree below this certificate.
- **-t:***types* is the certificate type. This parameter can be E for end-entity, C for certificate authority, or both.
- **-g** creates a glue certificate.
- **-r** creates a self-signed certificate.
- **-m** means to use the MD5 hash algorithm. This is the default.
- **-a** means to use the SHA1 hash algorithm.
- **-N** means to include Netscape client authentication extension.
- **-?** displays the options.

Here is an example:

```
>MakeCert -u:MyKey -n:CN=MySoftwareCompany Cert.cer
```

This generates a certificate file called Cert.cer. The public part of the key pair called KeyName is bound to the publisher, MySoftwareCompany.

This utility program should not be used once the software publisher obtains a valid X.509 software publisher

certificate from the appropriate CA.

Article ID: Q165831
Creation Date: 25-MAR-1997
Revision Date: 27-MAR-1997

The information in this article applies to:

® Microsoft Visual InterDev, version 1.0

Symptoms

Files will appear to be checked out or modified by the system's anonymous user when they are actually checked out to valid SourceSafe accounts.

Cause

The machine's Anonymous User account is a member of the Admin user group.

Visual SourceSafe will always attempt to perform actions as the anonymous user before trying to use the actual logged-in user's name. Only when an action exceeds the user rights allowed to the system's anonymous user will it attempt to authenticate as the actual user. Because the anonymous user is a member of the Admin group, it will always have the rights that are required to succeed in checking out files.

This can also be caused by using a Windows NT Server that has been formatted with FAT partitions as the Web server because, unlike drives formatted with NTFS, FAT drives have no direct way of securing files based on user id.

Resolution

Remove the anonymous user from the system's Admin group.

Status

This behavior is by design.

More Information

There are no known cases where a default installation would cause this situation. It was first discovered on a machine with manually modified rights.

The anonymous user mentioned here is an account name introduced by Microsoft Internet Information Server. It will have the format IUSR_<machinename>, and can be found in the Internet Service Manager under the properties for the World Wide Web Service.

Steps to Reproduce Behavior

1. Using the Windows NT User Manager, add the system's anonymous user to the group of Admin users.
2. Check out a file while logged in as a valid user.
3. Look in Visual SourceSafe on the server and you will see the files checked out to the anonymous user rather than the valid user.

Keywords : VIServer kbprb
Version : 1.0
Platform : WINDOWS
Issue type : kbprb

After you have generated a certificate, you must create an SPC (Software Publisher Certificate) with the Cert2SPC program. This program wraps multiple X.509 certificates into an SPC (the default) or a PKCS #7 signed-data object. The latter is simply an SPC with a PKCS #7 header added. Again, this program is for test purposes only. A valid SPC is obtained from a CA.

The syntax for Cert2SPC is:

Cert2SPC *[-7] cert1*.**cer** *cert2*.**cer**. . .*certN*.**cer** *output*.**spc**

where:

® **-7** means to prepend a PKCS #7 header to the SPC.

® *cert1*. . .*certN* are the names of the X.509 certificates to include in the SPC.

® *output* is the name of the SPC or PKCS #7 object containing the X.509 certificates.

Here is an example:

```
>Cert2Spc root.cer cert.cer cert.spc
```

This combines Cert.cer and Root.cer to make an SPC called Cert.spc.

The final step is to use the SPC to actually sign a file. This is done with the SignCode program. This program will:

1. Create a cryptographic digest of the file.
2. Sign the digest with your private key.
3. Copy the X.509 certificates from the SPC into a new PKCS #7 signed-data object. The PKCS #7 object contains the serial numbers and issuers of the certificates used to create the signature, the certificates, and the signed digest information.
4. Embed the object into the file.

If you have a valid SPC, then you can use this program to actually sign your code. The SignCode program has a wizard to help you do this. To sign code using the wizard, simply type "SignCode" without any options. If you want to sign your code manually, the syntax is:

SignCode [**-prog** *filename* **-spc** *credentials* **-pvk** *privateKeyFile*] [*options*]

where:

® **-prog** *filename* is the name of the file to sign.

® **-spc** *credentials* is the file that contains the credentials. This is usually an .spc file.

® **-pvk** *privateKeyFile* is the file containing the private key of the publisher. This is usually a .pvk file.

and where the *options* are:

® **-name** *opusName* is a name for your program.

® **-info** *opusInfo* is a location, such as an URL, for obtaining information about your program.

® **-gui** invokes the wizard.

® **-nocerts** means you do not want any X.509 certificates embedded in the PKCS #7 signed-data object. In this case, the relevant certificates must already be stored on the client computer.

® **-provider** *providerName* is the CryptoAPI provider to use. (See the CryptoAPI documentation for more details.)

® **-providerType** *providerType* is the CryptoAPI provider type to use. (See the CryptoAPI documentation for more details.)

® **-commerical** means the code being signed was created by a commercial software publisher.

® **-individual** means the code being signed was created by an individual software publisher. This is the default software publisher type.

® **-sha** means you want to use the SHA hashing algorithm.

® **-md5** means you want to use the MD5 hashing algorithm. This is the default hashing algorithm.

® **-?** displays the options.

Here is an example of how to sign a file:

```
>SignCode -prog MyProgram.exe -spc Cert.spc -pvk MyKey
```

This embeds a PKCS #7 object, Cert.spc, into the digest of file, MyProgram. The digest is signed with the private key of the MyKey key pair.

Once this is done (assuming you have a valid certificate), the file can be distributed to your customers.

The PeSigMgr program checks to see if SignCode was successful. This means the file should have a PKCS #7 object embedded in it. Here is the syntax:

PESIGMGR [*options*] *signedfile*

where the *options* are:

- **-l** lists the certificates in an image.
- **-a:**<-*filename* -> adds a certificate file to an image.
- **-r:**<-*index* -> removes certificate <-*index* -> from an image.
- **-s:**<-*filename* -> is used with **-r** to save the removed certificate.
- **-t:**<-*CertType* -> is used with **-a** to specify the type of certificate, where CertType may be X509 or PKCS7. (The default is PKCS7.)
- **-?** displays the options.

and where *signedfile* is the name of the signed file you want to check.

Here is an example:

```
>PeSigMgr -l MyProgram.exe
```

A sample response is:

```
>Certificate 0 Revision 256 Type PKCS#7
```

This means a certificate was embedded in the file (MyProgram.exe.)

The ChkTrust program checks the validity of the file. It does this by:

1. Extracting the PKCS #7 signed-data object.
2. Extracting the X.509 certificates from the PKCS #7 signed-data object.
3. Computing a new hash of the file and comparing it with the signed hash in the PKCS #7 object.

If the hashes agree, ChkTrust then verifies that the signer's X.509 certificate points back to the root certificate and that the correct root key was used.

If all these steps are successful, the file has not been tampered with, and the vendor was authorized to publish the file by the root authority.

Here is the syntax:

CHKTRUST [*type*] *signedfile*

where *type* is one of the following:

- **-s** indicates a software publishing trust provider. A dialog box is displayed for the untrusted case. This is the default setting.
- **-n** indicates a software publishing trust provider. No dialog box is displayed for the untrusted case. Instead, the HRESULT is displayed.
- **-w** indicates a Windows compatibility trust provider.
- **-c** is a cabinet file.
- **-i** is a PE image file.
- **-j** is a Java class file.

Here is an example:

```
ChkTrust MyProgram.exe
```

A successful response is:

```
Result: 0
```

The DumpCert utility is used to dump the certificates out of a certificate file, Software Publishing Certificate (SPC), or signed executable.

The syntax for invoking DumpCert is:

DUMPCERT *FileName*.*Extension*

where *Extension* is one of the following:

® **cer** indicates that the file is a certificate file containing one or more X.509 certificates.

® **spc** indicates that the file is a Software Publishing Certificate (SPC.)

® **exe** indicates that the file is a signed executable.

The result of the dump will appear similar to the following for each certificate:

Issuer:	"CN=Root Agency;"
Serial number:	02
Subject:	"1.3.6.1.4.1.311.2.1.25=Glue; CN=Test Time Stamp Root, O=Microsoft;"
Not before:	1996/10/30 16:22:22 (local time)
Not after:	1996/12/30 16:22:21 (local time)
Public key:	30 5B 30 0D 06 09 2A 86 48 86 F7 0D 01 01 01 05 00 03 4A 00 30 47 02 40 6B A1 F9 D1 46 F9 A9 CB 7E 49 2B B1 28 E1 6B 71 31 A3 59 0C 32 C6 64 7D DD B9 2A 8C 64 01 8B 52 79 1A B4 78 92 5D 26 69 2C F2 84 D2 8E 39 0A 91 F9 2E 5D D7 27 CA 98 9D 3C C6 E7 7D 03 0D D2 47 02 03 01 00 01
Public key hash:	9B CA AA 4D 0C 54 1D 70 4A 0A 1D FC 70 27 3C 8C
common name:	Test Time Glue Cert
auth key id: digest:	12 E4 09 2D 06 1D 1D 4F 00 8D 61 21 DC 16 64 63
iss/ser:	"CN=Root Agency;" F4 35 5C AA D4 B8 CF 11 8A 64 00 AA 00 6C 37 06
comm'l use:	yes (implicit)
ind'l use:	yes (implicit)
extensions:	2.5.4.3 2.5.29.1
Parent:	iss/ser: "CN=Root Agency;" F4 35 5C AA D4 B8 CF 11 8A 64 00 AA 00 6C 37 06 - signature ok

When creating an instance of the AdvancedDataFactory, include the prefix http:// or https:// before the server name.

When creating instances of your custom business objects, omit this prefix and specify only the server name.

The following articles are part of the "Client/Server Solution" series of articles written by Ken Bergman:

® The Architecture Process
® The Design Process
® The Basics
® Coding Guidelines
® Implementing the Layered Paradigm
® Cursors, Asynchronous Queries, and Handling Multiple Record Sets
® Leveraging SQL Server Services in a Transaction Processing Environment

---

Note These articles are all published on the January 1997 Microsoft Developer Network (MSDN) Library.

Ken Bergmann is the lead developer for the Microsoft Developer Network (MSDN). He became an expert in enterprise systems design by building them for Microsoft corporate use. Now he leads a team of developers who create samples detailing techniques and solutions for creating enterprise components. Proficient in Visual C++ and Visual Basic, Ken's emphasis is not on language-specific solutions, but on designing reusable, scaleable COM-based systems.

For some ActiveX controls, you must set a property to allow editing. For example, you must set the **AllowUpdate** and **AllowAddNew** properties of the data-bound Grid control to allow the user to edit and add data in the control.

To see a demonstration of the completed lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

You can jump directly to the lab exercises by clicking the titles below.

To see a demonstration of the completed lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

You can jump directly to the lab exercises by clicking the titles below.

To see a demonstration of the completed lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

You can jump directly to the lab exercises by clicking the titles below.

To see a demonstration of the completed lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

You can jump directly to the lab exercises by clicking the titles below.

To see a demonstration of the completed lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

You can jump directly to the lab exercises by clicking the titles below.

To see a demonstration of the completed lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

You can jump directly to the lab exercises by clicking the titles below.

To see a demonstration of the completed lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

You can jump directly to the lab exercises by clicking the titles below.

To see a demonstration of the completed lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

You can jump directly to the lab exercises by clicking the titles below.

To see a demonstration of the completed lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

You can jump directly to the lab exercises by clicking the titles below.

To see a demonstration of the completed lab solution, click this icon.
{ewc mvimg, mvimage,!democlip.bmp}

You can jump directly to the lab exercises by clicking the titles below.

You can view these code samples while reading the chapter, or you can click on the titles below.

{ewc MVIMG, MVIMAGE, !code.bmp} Lab Solution: Popup Menu Control <OBJECT> Tag

{ewc MVIMG, MVIMAGE, !code.bmp} Lab Solution: imgMath_Click()

You can view these code samples while reading the chapter, or you can click on the titles below.

{ewc MVIMG,MVIMAGE,!code.bmp} WebBot Validatation Function

{ewc MVIMG,MVIMAGE,!code.bmp} Error Handling

{ewc MVIMG,MVIMAGE,!code.bmp} Lab Solution: Scripting a Java Applet

{ewc MVIMG,MVIMAGE,!code.bmp}

Lab Solution: Scripting a Java Applet

{ewc MVIMG,MVIMAGE,!code.bmp}

Lab Solution: Scripting the Spin ActiveX Control

{ewc MVIMG,MVIMAGE,!code.bmp}

Lab Solution: Scripting the Reverse Button

{ewc MVIMG,MVIMAGE,!code.bmp}

Lab Solution: Creating a Hyperlink Object

{ewc

MVIMG,MVIMAGE,!code.bmp}

{ewc MVIMG,MVIMAGE,!code.bmp} Lab Solution: Creating a Hyperlink Object

{ewc MVIMG,MVIMAGE,!code.bmp} Lab Solution: Form Properties

{ewc MVIMG,MVIMAGE,!code.bmp} Lab Solution: Adding Form Validation

{ewc MVIM

G,
M
Vl
M
A
G
E,
!
co
de
.b
m
p}

You can view these code samples while reading the chapter, or you can click on the titles below.

{ewc_MVIMG_MVIMAGE,!code.bmp} Session_OnStart Redirect

{ewc_MVIMG_MVIMAGE,!code.bmp} Session_OnStart for Session Object Values

{ewc_MVIMG_MVIMAGE,!code.bmp} Browscap.ini

{ewc MVIMG, MVIMAGE, !code.bmp} Reading and Writing Text

{ewc MVIMG, MVIMAGE, !code.bmp} Lab Solution: Reading Form Data

{ewc MVIMG, MVIMAGE, !code.bmp} Lab Solution: Saving Session Information

{ewc MVIMG, MVIMAGE, !code.bmp} Lab Solution: Using Session Information

{ewc

MVIMG,MVIMAGE,!code.bmp}

{ewc MVIMG,MVIMAGE,!code.bmp} Lab Solution: Redirecting in Session_OnStart

{ewc MVIMG,MVIMAGE,!code.bmp} Lab Solution: Saving Requested Page

{ewc MVIMG,MVIMAGE,!code.bmp} Lab Solution: Redirecting to Requested Page

{ewc MVIM

G, MVlMAGE, !code.bmp}

{ewc MVlMG, MVlMAGE, !code.bmp}

Lab Solution: Incrementing a Hit Counter

{ewc MVlMG, MVlMAGE, !code.bmp}

Lab Solution: Saving Hit Counter to a File

{ewc MVlMG, MVlMAGE, !code.bmp}

Lab Solution: Reading Hit Counter from a File

{ewc MVlMG, MVl

MAGE,!code.bmp}

You can view these code samples while reading the chapter, or you can click on the titles below.

{ewc MVIMG, MVIMAGE, !code.bmp} Calling Parameterized Queries

{ewc MVIMG, MVIMAGE, !code.bmp} Error Handling by Redirecting

{ewc MVIMG, MVIMAGE, !code.bmp} Lab Solution: Creating a Connection

{ewc MVIMG, MVIMAGE, !code.bmp}

Lab Solution: Retrieve Transcript

{ewc MVIMG, MVIMAGE, !code.bmp}

Lab Solution: Creating a Table

{ewc MVIMG, MVIMAGE, !code.bmp}

Lab Solution: Printing Transcript Records

{ewc MVIMG, MVIMAGE, !code.bmp}

Lab Solution: Retrieving Parameters

{ewc

MVIMG,MVIMAGE,!code.bmp}

{ewc MVIMG,MVIMAGE,!code.bmp} Lab Solution: Creating a Connection

{ewc MVIMG,MVIMAGE,!code.bmp} Lab Solution: Opening the Feedback Table

{ewc MVIMG,MVIMAGE,!code.bmp} Lab Solution: Error Checking for Open Method

{ewc MVIM

G,
M
Vl
M
A
G
E,
!
co
de
.b
m
p}

{e
w
c
M
Vl
M
G,
M
Vl
M
A
G
E,
!
co
de
.b
m
p}

Lab Solution: ADC and DBList <OBJECT> Tags

{e
w
c
M
Vl
M
G,
M
Vl
M
A
G
E,
!
co
de
.b
m
p}

Lab Solution: Change DBList Field

{e
w
c
M
Vl
M
G,
M
Vl
M
A
G
E,
!
co
de
.b
m
p}

Lab Solution: ADS <OBJECT> Tag

{e
w
c
M
Vl
M
G,
M
Vl

MAGE,!code.bmp}

{ewc MVIMG,MVIMAGE,!code.bmp}

Lab Solution: Initialize ADC Controls

{ewc MVIMG,MVIMAGE,!code.bmp}

Lab Solution: DBList_Click()

{ewc MVIMG,MVIMAGE,!code.bmp}

You can view these code samples while reading the chapter, or you can click on the titles below.

{ewc MVIMG, MVIMAGE, !code.bmp} Lab Solution: Call Enrollment from .asp File

{ewc MVIMG, MVIMAGE, !code.bmp} Lab Solution: Drop Method

{ewc MVIMG, MVIMAGE, !code.bmp} Lab Solution: Transfer Method

You can view these code samples while reading the chapter, or you can click on the titles below.

{ewc MVIMG, MVIMAGE, !code.bmp} Lab Solution: Getting the Context Object

{ewc MVIMG, MVIMAGE, !code.bmp} Lab Solution: Calling CreateInstance

You can go directly to an animation or demonstration by clicking an icon or topic title below.

{ewc MVIMG,MVIMAGE,!anim.bmp}

Chapter 1 Introduction

{ewc MVIMG,MVIMAGE,!anim.bmp}

Web Site Architecture

{ewc MVIMG,MVIMAGE,!anim.bmp}

Web Site Development Team

You can go directly to an animation or demonstration by clicking an icon or topic title below.

{ewc MVIMG,MVIMAGE,!anim.bmp} Chapter 2 Introduction

{ewc MVIMG,MVIMAGE,!democlip.bmp} Creating a Project

{ewc MVIMG,MVIMAGE,!democlip Adding and Deleting Files

.bmp}

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Using Visual SourceSafe

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Adding Text and Images with the FrontPage Editor

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Using Script and Components

{ewc MVIMG,MVIMAGE,!an

im
.b
m
p}

{e
w
c
M
VI
M
G,
M
VI
M
A
G
E,
!
de
m
oc
lip
.b
m
p}

Viewing an Active Server Page

{e
w
c
M
VI
M
G,
M
VI
M
A
G
E,
!
de
m
oc
lip
.b
m
p}

Creating an Active Server Page

{e
w
c
M
VI
M
G,
M
VI
M
A
G
E,
!

Adding Standard Controls with FrontPage

democlip.bmp}

{ewc MVIMG, MVIMAGE, !democlip.bmp}

Using the Include Design-Time Control

{ewc MVIMG, MVIMAGE, !democlip.bmp}

Lab 2 Solution

You can go directly to an animation, demonstration, or expert point of view by clicking an icon or topic title below.

{ewc MVIMG,MVIMAGE,!anim.bmp} Chapter 3 Introduction

{ewc MVIMG,MVIMAGE,!democlip.bmp} Creating a Data Source

{ewc MVIMG,MVIMAGE,!democlip Adding a Data Connection

.bmp}

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Using Data View to Modify Data

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Modifying Database Structure

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Creating a Database Diagram

{ewc MVIMG,MVIMAGE,!de

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Using the Query Designer

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Using the Data Form Wizard

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Using the Data Range Controls

{ewc MVIMG,MVIMAG

E,
!
de
m
oc
lip
.b
m
p}

{e
w
c
M
VI
M
G,
M
VI
M
A
G
E,
!
de
m
oc
lip
.b
m
p}

Using a Parameter Query with Data Range Controls

{e
w
c
M
VI
M
G,
M
VI
M
A
G
E,
!
de
m
oc
lip
.b
m
p}

Lab 3 Solution

{e
w
c
M
VI
M
G,
M
VI
M
A
G
E,
!
de
m
oc
lip
.b
m
p}

Setting up the State University Database

{e
w
c
M
VI
M
G,
M
VI

MAGE,!democlip.bmp}

You can go directly to an animation or demonstration by clicking an icon or topic title below.

{ewc MVIMG,MVIMAGE,!anim.bmp} Chapter 4 Introduction

{ewc MVIMG,MVIMAGE,!democlip.bmp} Adding an ActiveX Control with Visual InterDev

{ewc MVIMG,MVIMAGE,!democlip} Adding an ActiveX Control with the FrontPage Editor

.bmp}

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Using a Licensed Control

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Creating and Using an HTML Layout

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Adding a Java Applet with the FrontPage Editor

{ewc MVIMG,MVIMAGE,!de

mocl ip.b mp}

{ew c MVI MG, MVI MA GE, !de mocl ip.b mp}

Lab 4 Solution

You can go directly to an animation or demonstration by clicking an icon or topic title below.

{ewc MVIMG,MVIMAGE,!anim.bmp} Chapter 5 Introduction

{ewc MVIMG,MVIMAGE,!democlip.bmp} Adding VBScript to a Web Page

{ewc MVIMG,MVIMAGE,!democlip Adding Event Procedures

.bmp}

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Using the Window_Onload Event

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Using FrontPage Validation

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Using the Script Wizard

{ewc MVIMG,MVIMAGE,!de

mocl
ip.b
mp}

{ewc MVIMG,MVIMAGE,!democlip.bmp} Using Frames Collection

{ewc MVIMG,MVIMAGE,!democlip.bmp} Using the Script Debugger

{ewc MVIMG,MVIMAG} Lab 5 Solution

E,
!
de
m
oc
lip
.b
m
p}

You can go directly to an animation or demonstration by clicking an icon or topic title below.

{ewc MVIMG,MVIMAGE,!anim.bmp} Chapter 6 Introduction

{ewc MVIMG,MVIMAGE,!democlip.bmp} Reading a Form

{ewc MVIMG,MVIMAGE,!democlip Using the Redirect Method

.b
m
p}

{e Posting Information to the Originating File
w
c
M
VI
M
G,
M
VI
M
A
G
E,
!
de
m
oc
lip
.b
m
p}

{e Using the Session Object
w
c
M
VI
M
G,
M
VI
M
A
G
E,
!
de
m
oc
lip
.b
m
p}

{e Using Events in Global.asa
w
c
M
VI
M
G,
M
VI
M
A
G
E,
!
de

mocl ip.b mp}

{ew c MVI MG, MVI MA GE, !de mocl ip.b mp}

Using the Browser Capabilites Component

{ew c MVI MG, MVI MA GE, !de mocl ip.b mp}

Lab 6 Solution

You can go directly to an animation, demonstration, or expert point of view by clicking an icon or topic title below.

{ewc MVIMG,MVIMAGE,!anim.bmp} Chapter 7 Introduction

{ewc MVIMG,MVIMAGE,!democlip.bmp} Overview of Using ADO and ADC

{ewc MVIMG,MVIMAGE,!exppov.b Data Access Models

{ewc MVIMG,MVIMAGE,!animbmp}

ActiveX Data Objects Architecture

{ewc MVIMG,MVIMAGE,!democlipbmp}

Creating a Connection Object

{ewc MVIMG,MVIMAGE,!democlip

Creating a Recordset

.bmp}

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Using the Command Object

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Handling Database Errors

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Lab 7.1 Solution

{ewc MVIMG,MVIMAGE,!de

mocliplbmp}

{ewc MVIMG,MVIMAGE,!animlbmp} Accessing Data Using ADC

{ewc MVIMG,MVIMAGE,!democliplbmp} Using the Advanced Data Control

{ewc MVIMG,MVIMAGE,! Using the Advanced Data Space

democlip.bmp}

{ewc MVIMG, MVIMAGE, !democlip.bmp}

Lab 7.2 Solution

You can go directly to an animation or demonstration by clicking an icon or topic title below.

{ewc MVIMG_MVIMAGE_!anim.bmp}

Chapter 8 Introduction

{ewc MVIMG_MVIMAGE_!democlii

Creating a Project with the Project Template

p
.
b
m
p
}

{
e
w
c
MVIMG
.
MVIMAGE
.
ani
m
.
b
m
p
}

Creating an Instance of a Class

{
e
w
c
MVIMG
.
MVIMAGE
.
ide

Adding a Method to a Class Module

mo c l i p . b m p }

{ Testing an ActiveX DLL

e w c _ M V I M G . _ M V I M A G E . ! d e m o c l i p . b m p }

{ Calling ActiveX Server Components

e w c _ M V I M G . _ M V

IMAGE_democlip.bmp}

{\ewc MVIMG_MVIMAGE_democlip.bmp}

Lab 8 Solution

You can go directly to an animation, demonstration, or expert point of view by clicking an icon or topic title below.

{ewc MVIMG,MVIMAGE,!anim.bmp} Chapter 9 Introduction

{ewc MVIMG,MVIMAGE,!expov.bmp} Microsoft Transaction Server

{ewc MVIMG,MVIMAGE,!anim.bm Microsoft Transaction Server

p}

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Creating a Package

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Adding a Component

{ewc MVIMG,MVIMAGE,!democlip.bmp}

Lab 9 Solution

{ewc MVIMG,MVIMAGE,!democ

lip
.b
m
p}

You can go directly to an animation, demonstration, or expert point of view by clicking an icon or topic title below.

{ewcMVIMG.MVIMAGE!animbmp}

Chapter 10 Introduction

{ewcMVIMG.MVIMAGE!democlipbmp}

Controlling Access to a Web Site

{

Lab 10 Solution

ewcMVIMG.MVIMAGE!.democlip.bmp}

{ewcMVIMG.MVIMAGE!.expposv.bmp}

Internet Security

{ew

Using Digital Certificates to Authenticate a User

cMVlMG_MVlMAGE_!_animbmp}

{ewcMVlMG_MVlMAGE_!_democlipbmp}

Using a Digital Certificate

You can view self-checks from within the chapter, or you can go directly to the self-check questions for Chapter 1 by clicking the title below.

Chapter 1: Planning a Web Site, Self-Check Questions

You can view self-checks from within the chapter, or you can go directly to the self-check questions for Chapter 2 by clicking the title below.

Chapter 2: Developing a Web Project, Self-Check Questions

You can view self-checks from within the chapter, or you can go directly to the self-check questions for Chapter 3 by clicking the title below.

Chapter 3: Using Visual InterDev Data Tools, Self-Check Questions

You can view self-checks from within the chapter, or you can go directly to the self-check questions for Chapter 4 by clicking the title below.

Chapter 4: Using Objects on Web Pages, Self-Check Questions

You can view self-checks from within the chapter, or you can go directly to the self-check questions for Chapter 5 by clicking the title below.

Chapter 5: Adding Client-Side Script, Self-Check Questions

You can view self-checks from within the chapter, or you can go directly to the self-check questions for Chapter 6 by clicking the title below.

Chapter 6: Using Active Server Pages, Self-Check Questions

You can view self-checks from within the chapter, or you can go directly to the self-check questions for Chapter 7 by clicking the title below.

Chapter 7: Creating Database-Aware Web Pages, Self-Check Questions

You can view self-checks from within the chapter, or you can go directly to the self-check questions for Chapter 8 by clicking the title below.

Chapter 8: Creating ActiveX Server Components, Self-Check Questions

You can view self-checks from within the chapter, or you can go directly to the self-check questions for Chapter 9 by clicking the title below.

Chapter 9: Using Microsoft Transaction Server, Self-Check Questions

You can view self-checks from within the chapter, or you can go directly to the self-check questions for Chapter 10 by clicking the title below.

Chapter 10: Controlling Access to a Web Site, Self-Check Questions

The Adventure Works sample application illustrates how an Internet business application can be created using Microsoft's Active Desktop and Active Server technologies:

® Microsoft Internet Explorer (IE)

® Microsoft Internet Information Server (IIS)

® Active Server Pages

® Microsoft Transaction Server

® ActiveX Data Objects(ADO)

® Advanced Data Connector (ADC)

® Microsoft SQL Server (ADC)

Adventure Works is a mythical store that sells outdoor adventure gear. The Adventure Works sample application is a web based online shopping system. It enables customers to browse through a catalog of products, select items to buy, and make purchases by supplying their name and credit card information. Adventure Works also performs merchant administrative tasks, such as maintaining customer lists, fulfilling orders, tracking inventory, and processing payments.

Click here to launch the Adventure Works sample application from the \SampApps\AdvWorks folder on the CD-ROM.
{ewc mvimg, mvimage,!exec.bmp}

The following sections contain more information:

® Installation Instructions

® Installation Problems

® Running Adventure Works

® Problems Running Adventure Works

® Building Adventure Works

Before Installing Adventure Works, you must install all of the following software:

1. **NT4.0 Service Pack 2.** Available from http://www.microsoft.com/NTServerSupport.
   {ewc mvimg, mvimage,!intjump.bmp}
2. **SQL Server 6.5 Service Pack 2.** Available from http://www.microsoft/SQLsupport.
   {ewc mvimg, mvimage,!intjump.bmp}
3. **Visual Basic 5 Professional or Enterprise Edition.**
4. **IIS 3.0, Active Server Pages, and Active Data Objects (ADO)**. Available from http://www.microsoft.com/iis.
   {ewc mvimg, mvimage,!intjump.bmp}
5. **Internet Explorer 3.01.** Available from http://www.microsoft.com/ie.
   {ewc mvimg, mvimage,!intjump.bmp}
6. **Advanced Data Connector.** Available from http://www.microsoft.com/adc.
   {ewc mvimg, mvimage,!intjump.bmp}
7. **Microsoft Transaction Server.** Available from http://www.microsoft.com/transaction.
   {ewc mvimg, mvimage,!intjump.bmp}

Installing Adventure Works on the Server

If you have completed the previous step, you may now install Adventure Works on the server. The installation program will ask you to select the directory where you wish to install the sample application files and it will then:

1. Copy the Adventure Works files to the directory you specify.
2. Create the AWData SQL database and insert sample data into the database tables.

Important Note The Adventure Works installation script assumes that SQL Server has been installed using the default SQL Server installation settings. For example, it assumes that the mssql\data directory exists and is located on the C: drive. If this directory does not exist, this portion of the installation will fail. However, you can correct this by modifying the portion of the MTxDemo\SQLScripts\AWCreate.SQL script that is shown below.

```
DISK INIT
  NAME = 'AWData',
  PHYSNAME = 'c:\mssql\data\AWData.dat',
  VDEVNO = @NextDeviceNumber,
  SIZE = 10238
GO
```

In the script, modify the "PHYSNAME" parameter to specify an existing directory in which the Adventure Works database should be created. Then run MTxDemo\MTXDATAVB5.BAT to create and populate the Adventure Works database.

3. Add the AWData Data Source Name (DSN) to ODBC.
4. Add the MtxDemo Virtual Directory to IIS.
5. Install the Adventure Works-Order Entry package into Microsoft Transaction Server (MTS).
6. Update the registry so that the Adventure Works components can be called from Active Server Pages.

To install Adventure Works on the server, click Install Adventure Works Now. When the dialog box is displayed, select Open it, click OK, then follow the instructions that appear on screen.

Installing Adventure Works on the Client

You may install the Adventure Works client on a separate machine. Note that this step is not required if you choose to run the client on the same machine as the server. To install the Adventure Works client on another machine, do the following:

1. Install IE3.01 on the client.

   You may obtain Internet Explorer 3.01 from http://www.microsoft.com/ie.

2. Copy the file Samples\AdventureWorks\AWClCntls.exe from this CD-ROM to a temporary directory on your

client machine. Then run AWCICntls.exe. This will install and register all of the necessary controls on the client machine.

If you have trouble installing Adventure Works please check the following:

1. Check for Required Software.

   Ensure that you have installed all of the required software. Failing to do this is the primary source of installation problems.

2. Install Microsoft Transaction Server after Microsoft SQL Server.

   Microsoft Transaction Server must be installed after SQL Server. If you install SQL Server after installing Microsoft Transaction Server, you must then reinstall Microsoft Transaction Server.

3. The Adventure Works installation script assumes that SQL Server has been installed using the default SQL Server installation settings. For example, it assumes that the mssql\data directory exists and is located on the C: drive. If this directory does not exist, this portion of the installation will fail. However, you can correct this by modifying the portion of the MTxDemo\SQLScripts\AWCreate.SQL script that is shown below.

   ```
   DISK INIT
     NAME = 'AWData',
     PHYSNAME = 'c:\mssql\data\AWData.dat',
     VDEVNO = @NextDeviceNumber,
     SIZE = 10238
   GO
   ```

   In the script, modify the "PHYSNAME" parameter to specify an existing directory in which the Adventure Works database should be created. Then run MTxDemo\MTXDATAVB5.BAT to create and populate the Adventure Works database.

4. You may also want to check the section, Problems Running Adventure Works.

Before Running Adventure Works

Before running Adventure Works, follow these steps:

1. Start SQL Server.
   a. Select **Start**.
   b. Select **Programs**.
   c. Select **Microsoft SQL Server 6.5**.
   d. Select **SQL Service Manager**.
   e. From the pull down menu select **MSSQLServer**.
      If MSSQLServer is not running, select **Start/Continue**.
2. Start Microsoft Distributed Transaction Coordinator.
   a. Select **Start**.
   b. Select **Programs**.
   c. Select **Microsoft SQL Server 6.5**.
   d. Select **SQL Service Manager**.
   e. From the pull down menu select **MSDTC**.
      If MSDTC is not running, select **Start/Continue**.
3. Start the IIS WWW Service.
   a. Select **Start**.
   b. Select **Programs**.
   c. Select **Microsoft Internet Server (Common)**.
   d. Select **Internet Service Manager**.
      If the WWW Service is not Running, double-click the **WWW Service**, then right-click your machine name and choose **Start**.
4. Set the IE3.01 security level to medium.
   a. Within Internet Explorer select **View**.
   b. Select **Options**.
   c. Select the **Security** tab.
   d. Select **Safety Level**.
   e. Select the **Medium** security method.
   f. Click **OK**, and then click **OK** again.
      Note that with the Safety Level set to medium, the following message will appear when you run Adventure Works:

      ```
      This page contains both scripts and a control called
      'grdProdGroup' that is not known to be safe for scripts
      to use. Do you want to allow the scripts on this
      page to access it?
      ```

      This message should not be appearing. You can circumvent this error by clicking **Yes to All**.
      You can also avoid the message by setting the Safety Level to **None**, but we strongly discourage this because it eliminates all browser security protection.

Running Adventure Works

With IE 3.01 running, type http://<your machine name>/MTxDemo in the tool bar address field and then press Enter.

Shopping in the Adventure Works Store

1. When the Adventure Works home page is displayed, select a Product Group by clicking on **Camping Equipment**, **Climbing Equipment**, or **Clothing**.
2. Select one of the four product types displayed in the **Select a Product Type** list box.
3. Select **Display Products**.
4. Add one of the products displayed on the catalog page to your Shopping Cart by clicking the **Order** button displayed next to the product.
5. When the Shopping Cart page is displayed, you may click **Shop for more** to select additional merchandise. You may enter a Color, Size, or Quantity and click **Recalculate** to compute the value of the goods in your shopping cart. You may click **Cancel Order** to discard the contents of your shopping cart. You may click **Place Order** to purchase the goods in your shopping cart.
6. When the Payment and Shipping page is displayed, you may search for an existing customer using the search facility on the left. You may then update the customer information using the form to the right. After you enter all of the credit card information, click **Place Your Order Now** to process the order.

   Note that the transaction is always aborted when the credit card type is **American Express**. This demonstrates what happens when a transactional component calls **SETABORT**.
7. When the order is successfully processed, the system will display the Congratulations page that contains the total cost of the order, the order number, and the authorization number.

Enrolling in an Adventure Works Expedition

To enlist for a climbing expedition, click **Base Camp** on the top toolbar. Select the excursion that you wish to attend, and click **Sign Up**. Use the form provided to enroll in the expedition of your choice.

Administering the Adventure Works Store

The Adventure Works store provides administrative capabilities for the Adventure Works merchant. By clicking **Inner Trail** on the top toolbar of the Adventure Works store, the Adventure Works merchant can review customer and order information.

The **Customer Listing** function provides customer address information such as name, address, and email name. If the merchant is using either the Microsoft Exchange or Outlook mail programs, clicking on the customer email name generates an email message addressed to that customer.

The **Sales By Customer** category provides a compilation of customers according to the volume of items purchased.

To review inventory activity, the merchant can select the **Sales By Product** option to learn which products have been ordered in what quantity.

The **Credit Check** option provides the merchant with the ability to verify that a customer has adequate credit for a payment method. An authorization number is also returned for tracking purposes.

If you have trouble running Adventure Works please check the following:

1. Ensure that you have installed all of the required software. Failing to do this is the primary source of problems when running Adventure Works.
2. Ensure that the security level in IE3.01 is set to **medium**.

   When Internet Explorer's Safety Level is set to **medium**, the following message appears:

   ```
   This page contains both scripts and a control called
   'grdProdGroup' that is not known to be safe for scripts
   to use. Do you want to allow the scripts on this
   page to access it?
   ```

   This message should not be appearing. You can circumvent this error by clicking **Yes to All**.

   You can also avoid the message by setting the Safety Level to **None**, but we strongly discourage this because it eliminates all browser security protection.
3. Ensure that the IIS WWW Service is running.
4. Ensure IIS is configured to allow anonymous logon.
5. Ensure that SQL Server is running.
6. Ensure SQL Server is configured to allow anonymous logon.

   If SQL Server and IIS are running on different machines, then the Anonymous user must be configured as a Guest user on the SQL Server machine.
7. Ensure that Microsoft Distributed Transaction Coordinator is running.
8. Ensure that the Adventure Works Order Entry Package is installed.

   To check whether the Order Entry package is installed:

   a. Select **Start**.

   b. Select **Programs**.

   c. Select **Microsoft Transaction Server**.

   d. Select **Transaction Server Explorer**.

   e. Select **My Computer**.

   f. Select **Packages Installed**.

   If the Adventure Works Order Entry package has not been installed:

   a. Select **File**.

   b. Select **New**.

   c. Select **Install pre-built packages**.

   d. Select **Add**.

   From the Add screen navigate to the <AW install dir>\ActiveXClient\ProgramFiles\VB5\Package\Aw.pak file and click on the file name.

   a. Select **Open**.

   b. Select **Next**.

   c. Select **Next**.

   d. Select **Finish**. This will install the pre-built Adventure Works Order Entry package.

Article ID: Q164586

The information in this article applies to:

® Microsoft OLE DB, version 1.1

Summary

This article describes how you can use ADO to access the sample text OLE DB provider.

More Information

The OLE DB SDK installs a sample text OLE DB provider that allows you to access text files. This sample text provider is located in the Samples\Sampprov directory. To use this sample text provider in ADO, you need to change the source code of the sample text provider, and you also need to change the ADO threading model. To do this, perform the following steps:

1. Modify the SetProperties function in the CutilProp class. The source file that contains this function is Utilprop.cpp. Change the last line of this function to return DB_E_ERRORSOCCURRED instead of E_FAIL. The complete line is:

   ```
   return ResultFromScode( DB_E_ERRORSOCCURRED  );
   ```

   ADO returns an error if the underline providers return E_FAIL when setting the OLE DB properties. The sample text driver needs to return DB_E_ERRORSOCCURRED instead of E_FAIL if there are any unsupported properties encountered.
2. The ADO threading model needs to be either "apartment" or "both." The default threading model is the apartment model. The sample text provider does not register any threading model, so it defaults to a single-threaded model. The threading models needs to match up in order for ADO to communicate with the sample text provider. After changing the threading model, you will need to restart Internet Information Server (IIS) for the changes to take effect.

You can use the batch file that came with the OLE DB SDK 1.1 to change the ADO apartment model. These files are located in the Bin\Util OLEDBSDK directory. The Makeapt batch file sets ADO to be apartment model threaded. You can create a REG file base on the following code and merge it with the registry to change the ADO threading model to both:

```
REGEDIT4

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0000022C-0000-0010-8000-
00AA006D2EA4}\InprocServer32]
"ThreadingModel"="Both"

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{00000231-0000-0010-8000-
00AA006D2EA4}\InprocServer32]
"ThreadingModel"="Both"

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{00000280-0000-0010-8000-
00AA006D2EA4}\InprocServer32]
"ThreadingModel"="Both"

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{00000281-0000-0010-8000-
00AA006D2EA4}\InprocServer32]
"ThreadingModel"="Both"

[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{00000293-0000-0010-8000-
00AA006D2EA4}\InprocServer32]
"ThreadingModel"="Both"
```

After you finish the two steps listed above, you can use the following Visual Basic code to use the sample text provider to retrieve data from a text file in ADO:

```
Private Sub myTest()
     Dim con As New ADODB.Connection
     Dim rs As ADODB.Recordset

     ' data source specifies path that contains customer.csv
     con.Open "provider=sampprov;data
source=c:\\sdks\\oledbsdk\\samples\\sampclnt\\"
     Set rs = con.Execute("customer.csv")

     For i = 0 To rs.Fields.Count - 1
         Print rs.Fields(i).Name
     Next i
     While Not rs.EOF
         Debug.Print rs(0)
         rs.MoveNext
     Wend
End Sub
```

Article ID: Q160226
Creation Date: 02-DEC-1996
Revision Date: 16-DEC-1996

The information in this article applies to:

® Microsoft FrontPage for Windows 97

Symptoms

When you uninstall FrontPage, Office 95, or Office 97, the proofing tools and text converters of the remaining application(s) may not work as expected.

Cause

Microsoft FrontPage 97 and Microsoft Office share the following components:

® Proofing Tools (Spelling Checker, Thesaurus)

® Text Converters

If FrontPage 97 is installed after Office 95, FrontPage will use the installed proofing tools. If you then uninstall Office 95, the proofing tools will be removed and will therefore be unavailable in FrontPage.

If FrontPage 97 is installed over Office 95, it will upgrade the existing text converters. If you then un-install FrontPage 97, it will remove the Msconv97.dll, rendering the remaining text converters useless to Office 95.

If Office 97 is installed before FrontPage 97, when Office 97 is un- installed, it removes the custom dictionary which means that the spelling checker no longer has access to it.

Workaround

To work around these problems, reinstall FrontPage 97 or the Proofing Tools and Converters components of Office.

Status

Microsoft has confirmed this to be a problem in FrontPage 97 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

KBCategory: kbinterop kbprb
KBSubcategory:
Additional reference words: 97

Article ID: Q85774
Creation Date: 18-JUN-1992
Revision Date: 02-APR-1997

The Application Note below, "Microsoft Download Service (MSDL) Instructions" (GA0604), describes the MSDL and gives instructions for downloading files and offers troubleshooting tips.

You can obtain this Application Note from the following sources:

® Microsoft's World Wide Web Site on the Internet

® The Internet (Microsoft anonymous ftp server)

® The Microsoft Network (MSN)

® Microsoft Download Service (MSDL)

® Microsoft FastTips Technical Library

® Microsoft Product Support Services

For complete information, see the "To Obtain This Application Note" section at the end of this article.

The Text of GA0604

Microsoft(R) Product Support Services Application Note (Text File)

GA0604: MICROSOFT DOWNLOAD SERVICE (MSDL) INSTRUCTIONS

Revision Date: 9/95

No Disk Included

What is the MSDL?

The MSDL operates like any MS-DOS-based computer bulletin board system (BBS). The MSDL contains Application Notes from Microsoft Product Support Services (PSS), as well as driver files and other types of support files for download. To use the MSDL, you must have a computer with a modem and a terminal package. Any terminal package, such as Microsoft Works, Windows Terminal, Procomm, or Crosstalk(TM), will work with the MSDL. If you experience difficulty while you are working with the MSDL, try calling a local BBS so you can avoid paying long- distance charges while you try to determine the cause of the problem.

Please note that technical support is not available for the MSDL.

How to Connect to the MSDL

The MSDL supports 1200, 2400, 9600, and 14,400 baud rates (V.32 and V.42), with 8 data bits, 1 stop bit, and no parity. Make sure the terminal software is configured to operate with these settings. After you have chosen these settings, you can begin the session as follows:

1. Call the MSDL at (206) 936-6735.
2. Enter your full name and the location you are calling from.

   The MSDL will list some basic instructions and information and then display the Main menu:

```
      ****************************************
      ****    Microsoft Download Service    ****
      ****             Main Menu            ****
      ****************************************

          [1] Download File
          [2] File Index (Find a file)

          [3] Instructions on Using This Service
          [4] Other Information & Options
      ----------------------------------------

          [L]ength of Call
          [E]xit - Logoff the System
          [H]elp - System Instructions

      Command:
```

How to Download Files

If you already know the exact name of the file to download, follow the steps below. If you do not know the filename, see the section below titled "How to Search for Files."

1. At the Main menu, press 1 for Download File.
2. Press D and then press the ENTER key.
3. When asked for the filename, type the name of the file you want to download. Be sure to include the correct extension (usually .exe); also, be sure to differentiate between the letter "O" and the number 0 (zero) in filenames.

Hint If you see a "-More-" prompt at the bottom of the screen, you can press the SPACEBAR to see more files.

---

4. When asked which protocol you would like to use, enter any protocol supported by your terminal package (check your software manual for more information). If you are unsure, press X for Xmodem.
5. The MSDL will then display the "Please start the download in your communications program" message. When you see this message, start the download process with your terminal software. For example, if you are using Windows Terminal, choose Receive Binary File from the Transfers menu.

   If you don't start the process, the transfer will fail, and you'll need to start again at step 2.
6. After your transfer is complete, you can quit the MSDL by pressing E to choose Exit.

How to Search for Files

If you do not know the exact name of the file to download, or if you simply want to find out what files are available, do the following:

1. At the Main menu, press 2 for File Index.
2. Press F for File Search.
3. You will be prompted to enter the text to search on. For example, if you want to search for a Hewlett-Packard(R) (HP(R)) LaserJet(R) printer driver, type "hp laserjet" (without the quotation marks) and press the ENTER key.

   The matching filenames will be displayed in a list.

Hint If you see a "-More-" prompt at the bottom of the screen, you can press the SPACEBAR to see more files.

If you don't find what you are looking for, see the following section, "How to Use the File Index," for instructions.

4. After you see the file that you want, write down the filename so you can download the file later.
5. Press the ENTER key to quit the search option.

For instructions about how to download the file, see the preceding section, "How to Download Files."

How to Use the File Index

1. At the Main menu, press 2 for File Index.
2. In the next screen (the File menu), choose the number that matches the application you are looking for.
3. Continue to select the appropriate number from the menus that appear until you see a list of files displayed on the screen.

   For example, if you are looking for a Windows 95 printer driver, do the following:

   a. At the Main menu, press 2 for File Index.
   b. Press 1 for Windows and MS-DOS.
   c. Press 1 for Windows 95 files.
   d. Press 4 for Windows 95 device drivers.
   e. Press 1 for Windows 95 printer drivers.
4. A list of files will be displayed on the screen. Read through the list to see if the file you need is displayed.

Hint If you see a "-More-" prompt at the bottom of the screen, you can press the SPACEBAR to see more files.

---

5. After you locate the file, write down the name. Press the SPACEBAR until you see the following at the bottom of the screen:

```
<D>ownload, <P>rotocol, <E>xamine, <N>ew, <L>ist, or <H>elp
Selection or <CR> to exit:
```

6. To download the file, press D, then follow steps 3-6 in "How to Download Files."

How to Use the File After It Is Downloaded

You now have your file. If the file has an .exe extension, run the file by typing its name at the MS-DOS command prompt. The .exe file will extract its contents and the resulting files are then ready to be installed. Please refer to any text files that have been extracted for exact instructions about using the downloaded file or files.

Customer Troubleshooting Guide for the MSDL

You may encounter one or more of the following problems when using the MSDL:

® Your connection fails.

® Your connection succeeds but results in garbage characters on the screen.

® The MSDL line constantly rings but does not answer.

® Your attempt to download a file results in the message "Please start the download in your communications program."

® Your attempt to download a file fails.

Procedures for correcting these problems follow, but please remember that if for some reason you cannot correct them, you can always receive MSDL files by downloading them, or by calling Microsoft Product Support Services at (206) 454-2030 (or your direct support number) or the Microsoft Sales Information Center (MSIC) at (800) 426-9400.

Your Connection Fails

If you cannot connect to the MSDL, or if you have connected but nothing happens, try the following steps in

order:

1. Check your communications protocol software and make sure data bits is set to 8, parity to none, and stop bits to 1--(8,N,1).
2. Select a lower baud rate. Doing this will result in slower data transfers, but it may allow you to get the files you need. If you still cannot connect or must use a higher baud rate, try step 3.
3. Disable modem data compression, specifically V.32bis and V.42bis, and try connecting again. There are different types of data compression, so make sure you disable all types on your modem.

   Data compression is intended to allow higher data transfer rates, but most MSDL files are already compressed. In fact, the transfer rate can be slower if your modem instructs the MSDL-side modem to compress files that are already compressed before sending them.

   To disable data compression, you must modify your terminal software Originate string. For example, most Courier modems require the addition of "&K0" to the Originate string. In Windows Terminal, choose Modem Commands from the Settings menu, and modify the Originate string ATQ0V1E1S0=0 to read ATQ0V1E1S0=0&K0. Data compression will be disabled the next time the modem is dialed.

   If you do not have a Courier modem, or if you are unable to determine exactly how to disable compression for your modem, see your modem manufacturer's documentation for instructions. If you disable all compression and still cannot connect to the MSDL, proceed to step 4.
4. Disable error correction. There are different types of error correction, so make sure you disable all levels. To disable error correction for Courier modems, add "&M0" to the terminal software Originate string. If you do not have a Courier modem, see your modem manufacturer's documentation for instructions.
5. If none of these steps corrects the problem, try a different modem.

If the connection still fails, obtain the files by one of the other means available to you. If you need further assistance, call Microsoft Product Support Services at (206) 454-2030 (or use your direct support number).

Your Connection Succeeds but Results in Garbage Characters on the Screen

1. Make sure the number you are dialing is (206) 936-6735.
2. Make sure that data bits are set to 8, parity to none, and stop bits to 1 (8,N,1) in your communications program.
3. Use the troubleshooting tips under the section titled "Your Connection Fails."

The MSDL Line Constantly Rings but Does Not Answer

If calling the MSDL results in constant ringing, try calling again later in the day. This problem is caused by the failure of one or more modems to reset correctly after a user has hung up. Generally this problem occurs when the MSDL is extremely busy. This situation is usually corrected soon after it is detected.

Your Attempt to Download a File Results in the Message "Please Start the Download in Your Communications Program"

To correct this problem, you must start the Receive File process in your communications software. For example, in Windows Terminal, choose Receive Binary File from the Transfers menu. The MSDL cannot send data until it receives this signal.

This information is also listed in the MSDL Main Menu under Help.

Your Attempt to Download a File Fails

If your download attempt fails, try these procedures:

1. Make sure you have selected the same protocol in your communications software and from the MSDL.
2. Switch to a different protocol.
3. Disable data compression on your modem. (See step 3 in the "Your Connection Fails" section.)
4. Try a lower baud rate or connect to the MSDL again. This often corrects problems caused by bad phone

connections or noisy phone lines.

If the download still fails, obtain the files by one of the other means available to you. If you need further assistance, call Microsoft Product Support Services at (206) 454-2030 (or use your direct support number).

To Obtain this Application Note

You can find GA0604.EXE (size: 17685 bytes), a self-extracting file, on the following services:

® Microsoft's World Wide Web Site on the Internet

1. On the www.microsoft.com home page, click the Support icon.
2. Click Knowledge Base.
3. Select the product.
4. Enter kbfile GA0604.EXE (size: 17685 bytes).
5. Click GO!
6. Open the article, and click the button to download the file.

® Internet (anonymous FTP)

1. ftp ftp.microsoft.com
2. Change to the Softlib/Mslfiles folder.
3. Get GA0604.EXE (size: 17685 bytes)

® The Microsoft Network

1. On the Edit menu, click Go To, and then click Other Location.
2. Type "mssupport" (without the quotation marks).
3. Double-click the MS Software Library icon.
4. Find the appropriate product area.
5. Locate and Download GA0604.EXE.

® Microsoft Download Service (MSDL)

1. Dial (206) 936-6735 to connect to MSDL
2. Download GA0604.EXE (size: 17685 bytes)

For additional information about downloading, please see the following article in the Microsoft Knowledge Base article:

ARTICLE-ID: Q119591

TITLE: How to Obtain Microsoft Support Files from Online Services

You can have this Application Note mailed or faxed to you from the automated Microsoft FastTips Technical Library, which you can call 24 hours a day, 7 days a week at (800) 426-9400. NOTE: The FastTips Technical Library is available only to customers within the U.S. and Canada.

If you are unable to access the sources listed above, you can have this Application Note mailed or faxed to you by calling Microsoft Product Support Services Monday through Friday, 6:00 A.M. to 6:00 P.M. Pacific time at (206) 454-2030 (or use your direct support number). If you are outside the United States, contact the Microsoft subsidiary for your area. To locate your subsidiary, see the Microsoft World Wide Offices Web site at http://www.microsoft.com/worldwide/default.htm

Additional reference words: WDL WADLE tshoot appnote ga0604 download
KBCategory: kbref kbtshoot kbappnote kbfile
KBSubcategory:
Keywords : kbappnote kbfile kbref kbtshoot

The goal of the Adventure Works sample application is to show you how Microsoft Transaction Server and Microsoft's other Active Desktop and Active Server technologies can be used to write Internet and Intranet applications. We have designed Adventure Works to incorporate as many of these technologies as possible, while still keeping the application realistic. Applications you build are unlikely to incorporate so many of these technologies in one application. For example, you might choose to use Active Server Pages and Microsoft Transaction Server on the server but not use ActiveX controls or VB Script on the client. It is up to you to decide which technologies best fit your application. Adventure Works shows you what is possible when you combine all of these powerful technologies in one application.

The following topics are included in this section:

To really understand the application, you will need to read the code. All of the source code for Adventure Works is included on the *Mastering Web Site Development* CD-ROM, in the \SampApps\AdvWorks folder.

The Sample Bank application illustrates how easy it is to build a Microsoft Transaction Server application. Sample Bank is a three tier, transactional, database application that can debit or credit a single bank account or transfer money between two accounts. It features:

1. A client written in Visual Basic
2. Microsoft Transaction Server business components written in Visual Basic, Visual C++, and J++. By replicating these components in these three languages, we allow you to compare how the same business logic can be implemented using each language.
3. A SQL Server database that is updated under transaction control.

The following sections provide instructions for installing and running the Sample Bank application:

® Installing Bank

® Running Bank

® Running Bank on a Remote Client

® Building Bank

To install Sample Bank on the Server:

1. On the **Start** menu:
   a. Point to **Programs**.
   b. Point to **Microsoft Transaction Server**.
   c. Select **Transaction Server Explorer**.
2. In the left pane:
   a. Double-click the **My Computer** icon.
   a. Double-click the **Packages Installed** folder.
3. On the **File** menu:
   a. Click **New**.
   b. Click the **Install pre-built packages** button.
   c. Click **Add**.
   d. Select **Sample Bank.PAK** in the **..\Samples\Packages** folder of your Microsoft Transaction Server installation.
   e. Click **Open**.
   f. Click **Next**.
4. In the **Set Package Identity** dialog box:
   a. Select **Interactive user**.
   b. Click **Next**.
5. In the **Installation Options** dialog box, specify the installation directory, for example, **C:\Mtx\Packages**.
6. Click **Finish**.

Configuring the ODBC Data Source

Use the ODBC applet in Control Panel to configure your Sample Bank data source name.

1. On the **Start** menu:
   a. Point to **Settings.**
   b. Select **Control Panel**.
2. In the Control Panel:
   a. Double-click **ODBC**.
3. In the **ODBC Data Source Administrator** dialog box:
   a. Click **System** DSN.
   b. Click **Add**.
   c. Select **SQL Server**.
   d. Click **Finish**.
   e. Specify the data source name as **MTxSamples** and click **Options**.
   f. Specify the data source name as **MTxSamples** and the Server as (**local**).
   g. Click **Options** and enter the default database to use for the Account table (**'pubs'** for example).

Preparing to Run the Sample Bank Application

You can monitor the execution of the Sample Bank application using the Microsoft Transaction Server Explorer. To do this:

1. On the **Start** menu:
   a. Point to **Programs**.
   b. Point to **Microsoft Transaction Server**.
   c. Select **Transaction Server Explorer**.
2. In the left pane:
   a. Double-click the **My Computer** icon.
   b. Double-click the **Packages Installed** folder.
   c. Double-click the **Sample Bank package** icon in the left pane of the Microsoft Transaction Server Explorer.
   d. Double-click the **Components** folder.
3. On the **View** menu:
   a. Click **Status** to display usage information for the various components in the package.
   b. Click **New Window**.
   Re-arrange the new window so it does not overlap.
   a. Click **Transaction Statistics** in the left pane of the new window.
4. On the **View** menu:
   a. Click **Hierarchy** to hide the left pane.
   Transactions statistics will be displayed when transactional components are used.

Running the Sample Bank Application

To run the Sample Bank application:

1. Start SQL Server.
   a. Select **Start**.
   b. Select **Programs**.
   c. Select **Microsoft SQL Server 6.5**.
   d. Select **SQL Service Manager**.
   e. From the pull down menu, select **MSSQLServer**.
   If MSSQLServer is not running, select **Start/Continue**.
2. Start Microsoft Distributed Transaction Coordinator.
   a. Select **Start**.
   b. Select **Programs**.
   c. Select **Microsoft SQL Server 6.5**.
   d. Select **SQL Service Manager**.
   e. From the pull down menu, select **MSDTC**.
   If MSDTC is not running, select **Start/Continue**.
3. Start the Sample Bank Client.
   a. On the Start menu, point to **Programs**.
   b. Point to **Microsoft Transaction Server**.
   c. Point to **Samples**.
   d. Select **Bank Client**.
   Arrange the Bank Client window so that it does not overlap the Microsoft Transaction Server Explorer windows.

The form will default to credit $1 to account number 1.

e. Click **Submit**.

You should see a response with the new balance.

4. Observe the Microsoft Transaction Server Explorer windows. You will notice that the component usage and transaction statistics windows have been updated.

Experiment with the bank client and observe the statistics using different business transaction types and varying numbers of iterations. You might notice that the very first transaction takes longer than the others. This is because the first transaction creates the sample bank database tables and inserts temporary records into them.

To install the Sample Bank client application on a remote client machine, do the following:

1. Copy the following files from the **Samples\Account.VB5\RemoteClientReg** directory on the CD-ROM to the client machine:

   ```
   Clireg32.EXE
   EnableDCOM.REG
   Vbacct.TLB
   Vbacct.VBR
   VbacctReg.BAT
   ```

2. Run **VbacctReg.BAT** on the client machine.
3. When the **Configuration Dialog** is displayed, enter the name of the server where the Sample Bank component is installed in the **Network Address** text box.
4. Click the **OK** button.

   Doing this will cause the client to be registered.
5. You may then run the Sample Bank client application.

   To learn how to run Sample Bank, see Running Bank.

RemoteClientReg Directory Contents

® Vbacct.TLB

Vbacct.TLB is the type library for the component. You may copy either Vbacct.DLL or Vbacct.TLB to the client. Both files contain exactly the same type library information. However, the Vbacct.TLB file is significantly smaller because it contains only the type library information.

® **Vbacct.VBR**

The Vbacct.VBR file is much like a registration file. Clireg32.EXE reads it.

® **EnableDCOM.REG**

The EnableDCOM.REG file is a registration file that is processed by REGEDIT.EXE. It ensures that DCOM is configured on the client machine. It contains the following commands:

```
REGEDIT4
; Ensure that DCOM is configured to allow access
; to Microsoft Transaction Server

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Ole]
"EnableDCOM"="Y"
"LegacyAuthenticationLevel"=dword:00000004
"LegacyImpersonationLevel"=dword:00000002
```

® **Clireg32.EXE**

The Clireg32.EXE program.

® **VbacctReg.BAT**

The VbacctReg.BAT file is a command file that invokes REGEDIT and Clireg32. If you wish to use it to register a different components, you must modify it to reflect your component's name. This file contains the following commands:

```
@ECHO OFF
REGEDIT.EXE -s EnableDCOM.REG
Clireg32.EXE Vbacct.VBR -d -nologo -t Vbacct.TLB
```

Creating TLB and VBR Files using Visual Basic

You can create Vbacct.TLB and Vbacct.VBR files using Visual Basic 5.0 as follows:

1. Start Visual Basic 5.0.

2. Select the **Existing** tab and open ..**Samples\Account.VB\Account.vbp**.
3. On the **Project** menu:
   a. Select the **Bank Properties** menu item.
   b. Select the **Component** tab.
4. Set the **Remote Server Files** checkbox.
5. Set the **Binary Compatibility** option and ensure that the package name is **..\..\Packages\Vbacct.dll**.
6. On the **File** menu, select the **Make Vbacct.dll** menu item.
7. Compile **Vbacct.dll** into an empty directory.

Important Note Do not overwrite the existing **Vbacct.dll** file.

---

8. VB will create the **Vbacct.TLB** and **Vbacct.VBR** files in the directory you specified.

You can choose to install the source, object, and project files for building the Sample Bank application when you install Microsoft Transaction Sever. If you choose this installion option, the files for the Sample Bank application will be installed in the \Samples folder of your Microsoft Transaction Server installation directory.

Before you open the **..\Samples\Bank.VB\Bank.vbp** project file using Visual Basic 5 you should install Sample Bank. If you do not do this, you will get the warning message "Unable to set the version compatible component: ..\Mtx\Packages\Vbacct.dll". If you encounter this warning, simply install Sample Bank.

If you open either **..\Samples\Account.VC\Account.mdp** or **..\Samples\Account.VJ\Account.mdp** using Visual Studio 97 you may get the warning "This project was generated by a previous version of Developer Studio. Continuing will convert it to the new format." You may continue by responding **Yes** when asked if you wish to convert it to the new format.

Sample Element	File Location
VB 5.0 Client Source Code	..\Samples\Bank.VB\Bank.vbp
VB 5.0 Server Source Code	..\Samples\Account.VB\Account.vbp
VC++ 4.2 Server Source Code	..\Samples\Account.VC\Account.mdp
J++ 1.0 Server Source Code	..\Samples\Account.VJ\Account.mdp
All Object Code Files	..\Samples\Packages

Part II of the *Microsoft Transaction Server Programmer's Guide* contains step by step instructions for creating, installing, and running the Sample Bank Account component in Visual Basic. You can find the Microsoft Visual Basic projects for each of these steps in the Step1 through Step8 folders in the **..\Samples\Account.VB** folder on this CD-ROM. You can use the Windows Explorer to copy these files from the CD-ROM to your local drive.

If you have trouble running the Sample Bank application please check the following:

1. Ensure that SQL Server is running.
2. Ensure that Microsoft Distributed Transaction Coordinator is running.
3. Ensure that the Sample Bank Package is installed.

To check whether the Sample Bank package is installed:

1. Select **Start**.
2. Select **Programs**.
3. Select **Microsoft Transaction Server**.
4. Select **Transaction Server Explorer**.
5. Select **My Computer**.
6. Select **Packages Installed**.

If the Sample Bank package has not been installed, follow the steps described in Installing Bank.

{ewc mvimg, mvimage,!illust.bmp}

The Adventure Works application uses many of the Active Desktop and Active Server technologies including:

® Internet Explorer (IE)

The Internet Explorer browser provides the Adventure Works application's user interface. Internet Explorer displays web pages constructed from HTML. It hosts ActiveX controls. It supports client-side scripts that control the elements on the page.

® Internet Information Server (IIS)

The Internet Information Server receives and responds to HTTP requests. It hosts both Active Server Pages and the Advanced Data Connector server.

® Active Server Pages

Active Server Pages combines HTML and server side scripting to generate dynamic HTML. Adventure Works uses server side scripting to generate dynamic HTML. It also uses server side scripting to invoke Microsoft Transaction Server business components.

® Microsoft Transaction Server

Microsoft Transaction Server provides the runtime environment in which the Adventure Works application business components execute.

® Advanced Data Connector (ADC)

The Advanced Data Connector permits a client on one machine to read and update a SQL database on another machine. The client and server systems may be connected via HTTP, HTTPS, or DCOM. ADC is responsible for transmitting SQL statements and SQL data between the client and server machines. ADC permits the client to request a set of rows from the server. ADC caches these rows on the client to improve performance. This client may update these cached rows. It may then instruct ADC to send the modified rows to the server to update the database.

ADC's Advanced Data Space facility can be used to invoke business components via HTTP, HTTPS, or DCOM. The invoked component can return a rowset. ADC can bind this rowset to one or more client-side ActiveX controls.

® Microsoft SQL Server

Microsoft SQL Server stores the Adventure Works application's persistent data.

® ActiveX Data Objects (ADO)

ActiveX Data Objects provides universal data access. It provides simple interfaces for reading and writing data from a wide variety of data sources including: relation databases, desktop databases, file systems, office applications, mail systems, and other data sources. Adventure Works use ADO to read and update business data stored in a Microsoft SQL Server database.

{ewc MVIMG, MVIMAGE,!2034_02.bmp}

The Internet Explorer browser provides the Adventure Works application's user interface.

Adventure Works uses HTML to control the layout of information on web pages. It exploits frames and ActiveX controls, such as list boxes and buttons, to provide a richer graphical user interface.

Adventure Works uses hidden ActiveX controls, such as the ADC client component. This ADC client control is used to retreive information from a SQL database on the server. The hidden ADC client control is bound to a visible ActiveX list box control that displays the types of products available for sale.

Adventure Works uses client-side Visual Basic scripts to control the elements on the page. For example, when the page is first loaded, a Visual Basic script performs a SQL query using the ADC client control. The results of this query are used to populate the list box showing the types of available products. Other Visual Basic scripts are invoked when list items are selected or buttons are clicked.

Click here to connect to the Microsoft Internet Explorer Web site for more information.
{ewc mvimg, mvimage,!intjump.bmp}

The Internet Information Server (IIS) receives and responds to HTTP requests. It manages connections, provides directory services for managing Web pages, maintains logs, and enforces security. It hosts other Active Server components including Active Server Pages and the Advanced Data Connector server.

Click here to connect to the Microsoft Internet Information Server Web site for more information.
{ewc mvimg, mvimage,!intjump.bmp}

Active Server Pages combines HTML and server side scripting to generate dynamic HTML. Active Server Page scripts can also invoke a variety of standard components. These components can be used to determine the capabilities of the browser, to access conventional files, and to select among a list of potential online advertisements, for example. Active Server Pages scripts can read and update SQL databases through ADO. Finally, Active Server Page scripts can invoke COM and Microsoft Transaction Server components.

Adventure Works uses Active Server Pages scripts to generate dynamic HTML and to invoke Microsoft Transaction Server business components.

Click here to connect to connect to the Microsoft Web site to learn more about Active Server Pages.
{ewc mvimg, mvimage,!intjump.bmp}

{ewc mvimg, mvimage,!illust.bmp}

Microsoft Transaction Server is a component-based transaction processing system for developing, deploying, and managing high performance, scalable, and robust enterprise, intranet and Internet, server applications. Transaction Server defines an application programming model for developing distributed, component-based applications. It also provides a run-time infrastructure for deploying and managing these applications.

Base clients are the primary consumers of Microsoft Transaction Server components. An Active Sever Page is an example of a base client. The Transaction Server components implement the business rules using Visual Basic, Visual C++, Visual J++, or any ActiveX-compatible development tool.

Transaction Server allows developers to focus on implementing business functions rather than on complex server issues. Because components running under Transaction Server can take advantage of transactions, developers can write applications as if the programs run in isolation. Transaction Server handles the concurrency, resource pooling, security, context management, and other system-level complexities. The transaction system, working in cooperation with databases, such as Microsoft SQL Server, ensures that concurrent transactions are atomic, consistent, have proper isolation, and that, once committed, the changes are durable.

Click here to connect to the Microsoft Transaction Server Web site for more information.
{ewc mvimg, mvimage,!intjump.bmp}

Microsoft Advanced Data Connector (ADC) is a high-performance, Web-based technology that brings plug-and-play database connectivity and corporate data publishing capabilities to Internet and intranet applications.

With Microsoft Advanced Data Connector (ADC), you can build intelligent Web applications that let you access and update data from an ODBC-compliant database. And because you implement ADC with familiar technology - off-the-shelf visual controls, HTML, and Microsoft Visual Basic Scripting Edition (VBScript) - ADC integrates seamlessly with existing Visual Basic applications, letting you transport them to the Web.

A key feature of Microsoft Advanced Data Connector is a client-side caching mechanism that minimizes connections to your database. As a result, you get better performance than in traditional Web database access methods. Such performance improvements are especially noticeable when accessing data across the Internet.

ADC's Advanced Data Space facility can be used to invoke business components via HTTP, HTTPS, or DCOM. The invoked component can return a rowset. ADC can bind this rowset to one or more client-side ActiveX controls.

Click here to connect to the Microsoft Advanced Data Connector Web site for more information.

Microsoft SQL Server, a relational database management system for distributed client/server computing, is designed to support large-scale distributed computing environments. It includes logging and transaction facilities to safeguard your critical data.

Adventure Works uses Microsoft SQL Server to store persistent application data.

Click here to connect to the Microsoft SQL Server Web site for more information.
{ewc mvimg, mvimage,!intjump.bmp}

ActiveX Data Objects provides universal data access. It provides simple interfaces for reading and writing data from a wide variety of data sources including: relation databases, desktop databases, file systems, office applications, mail systems, and other data sources.

Adventure Works uses ADO to read and update business data stored in Microsoft SQL Server.

Click here to connect to the Microsoft ActiveX Data Objects Web site for more information.
{ewc mvimg, mvimage,!intjump.bmp}

{ewc mvimg, mvimage,!illust.bmp}

This section provides a description of the following:

® Three Tier Architecture
® Adventure Works Application Components
® Adventure Works Database
® Adventure Works Files

This section describes how to:

® Use ADO
® Use ADC
® Pass a Rowsets as a Parameter
® Invoke an MTS Component
® Download and Use an ActiveX Control
® Use the Shared Property Manager

The Adventure Works application is composed of three logical tiers: a Client Tier that implements the user interface, a Middle Tier that implements the business logic, and a Data Tier that stores the application's persistent data. These three logical tiers do not necessarily correspond to how the application is physically distributed. For example, a single logical tier could be distributed over two or more computers, or two logical tiers could reside on a single computer.

{ewc MVIMG, MVIMAGE,!2034_07.bmp}

Client Tier

The client tier implements the application's graphical user interface. It is responsible for displaying information to the user and getting input from the user. The client tier may consist of browser displayed web pages, or of forms displayed by programs written in Visual Basic, Visual C++, J++, Powerbuilder, Delphi, and the like.

The client tier is usually responsible for validating user entered data. For example, it might ensure that a date field is specified in dd/mm/yy format and is within a specifed range.

Middle Tier

The middle tier implements the application business rules. For example, it might enforce the rule that every customer order must contain at least one order item and that the customer's account must be in good standing when an order is placed.

The middle tier is also responsible for controlling the flow of work through the application. For example, an insurance application might ensure that a customer's claim has been validated before it is paid.

Data Tier

The data tier stores the application's persistent data. This data may be stored in relational databases, desktop databases, conventional files, office documents, mail folders, image stores, voice stores, and the like.

Critical business data is usually transaction protected. Most relational databases support transactions and other data stores are being extended to support them.

Adventure Works is composed of the following application components:

® AuthorizePayment
® Catalog
® Customers
® Order
® Payment
® SalesTax
® Shipping
® ShoppingBasket
® ShoppingBasketItem
® TakeANumber
® TakeANumberUpdate

The AuthorizePayment component is responsible for checking whether a credit card charge is authorized.

The AuthorizePayment method is passed the customer's credit card information and the purchase amount. If the payment is authorized, AuthorizePayment returns a credit authorization number. Otherwise it returns zero.

Attributes

ProgID: AWAuthorize.AuthorizePayment
Project file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\PayAuthorization\AWAuth.vbp
DLL file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Package\AWAuthorize.dll
Transaction: Supports Transactions

Public Methods

AuthorizePayment(lngPaymentMethod As Long, strCardNumber As String, strCardName As String, dblCardAmount As Double, datCardExpire As Date) As Long

Where Used

Payment

The Catalog component is responsible for reading one or more records from the Product table.

The GetCatalogItemByCode method returns the single Product record with the specified Product Code.

The GetCatalogItemByProductType method returns the set of Product records of the specified Product Type.

Attributes

ProgID: AWCatalog.Catalog
Project file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Cat\AWCat.Vbp
DLL file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Package\AWCat.dll
Transaction: Does Not Support Transactions

Public Methods

GetCatalogItemByCode(ByVal vProductCode) As ADOR.Recordset

GetCatalogItemsByProductType(ByVal vProductType) As ADOR.Recordset

Where Used

catalog_type.asp
checkout.asp
submit_order.asp

The Customers component is responsible for reading and updating customer records.

The LookupCustomerByLastName method returns the set of Customers records that match the specified portion of the last name.

The LookupCustomerByID method returns the Customers record that matches the specified Customer ID.

The UpdateCustomers method updates the Customers records that the caller passes via the record set.

Attributes

ProgID: AWCustomer.Customers
Project file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Cust\AWCust.vbp
DLL file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Package\AWCust.dll
Transaction: Requires a Transaction

Public Methods

LookupCustomerByID(ByVal vCustomerID) As ADODB.RecordSet

LookupCustomerByLastName(ByVal vCustomerLastName) As ADODB.RecordSet

UpdateCustomers(adoWork As ADODB.Recordset)

Where Used

CustInfoNew.asp

The Order component is responsible for processig the customers order. It is the central component of the Adventure Works application.

The GetSalestax method determines the amount of sales tax for the order by calling the SalesTax component's Salestax method.

The Submit method is the most important method in the Adventure Works application. It is responsible for controlling the ordering process. It first calls AddHeaderToOrder to create the order header record in the Orders table. It calls the Shipping Component's SetHeader method to create the shipping header. It calls AddItemsToOrder to create an OrderDetails record for each item the customer has placed in thier shopping basket. It calls the AddPayment method to process the payment. It calls the Shipping component's Ship method to finalize the shipment. Finally it returns the Order Identifier, the Payment Identifier, and the charges associated with the order.

The AddPayment method calls the Payment component's PostPayment method to process the payment.

The AddHeaderToOrder method inserts a new record in the Orders table. This record represents the new order.

The UpdateInventory method decrements the QuantityOnHand field in the appropriate Inventory database record.

The AddItemsToOrder method iterates through the ShoppingBasketItems in the ShoppingBasket collection. For each ShoppingBasketItem contained in the collection, it first inserts a record in the OrderDetails table. It then calls the Shipping component's AddItem method to generate a shipping line item.

The private GetOrderID method calls the TakeANumber Component's GetAscendingFast method to generate a unique order identifier.

Attributes

ProgID: AWOrder.Order
Project file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Order\AWOrder.Vbp
DLL file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Package\AWOrder.dll
Transaction: Requires a Transaction

Public Methods

AddHeaderToOrder()

AddItemsToOrder(objBasket As AWBasket.ShoppingBasket, objShipping As AWShipping.Shipping)

AddPayment(lngOrderID As Long, strCreditCard As String, strCardName As String, strCardNumber As String, datCardExpire As Date) As Long

GetSalesTax(strStateOrProvince As String) As Currency

Submit(ByVal objBasket As AWBasket.ShoppingBasket, vCustomerID As Long, vStateOrProvince As String, vCreditCardType As String, vCreditCardNumber As String, vCreditCardName As String, vCreditCardExpire As Date) As Variant

UpdateInventory(vProductCode As String, vQuantity As Double)

Private Methods

GetOrderID() As Long

Where Used

submit_order.asp

When an order is processed, the Payment component is responsible for verifying the customer's credit card information and recording the charge against the customer's credit card account.

The PostPayment method takes a customer’s credit card information as input and calls the AuthorizePayment component's AuthorizePayment method to determine whether the credit card is valid. AuthorizePayment returns an Authorization Number if the card is valid and zero otherwise. If the card is valid, the PostPayment method calls the private GetPaymentID method to allocate the next Payment Identifier. PostPayment then inserts a record in the Payments table which represents the charge against the customer's credit card. Finally, PostPayment returns the Authorization Number to its caller thereby indicating that the payment was processed successfully. PostPayment returns zero if the payment could not be processed.

The private GetPaymentID method generates a Payment Identifier by calling the TakeANumber component's GetAscendingFast method. It returns this Payment Identifier to its caller.

Attributes

ProgID: AWPayment.Payment
Project file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Payment\AWPayment.Vbp
DLL file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Package\AWPayment.dll
Transaction: Requires a Transaction

Public Methods

PostPayment(lngOrderID As Long, strCardType As String, strCardNumber As String, strCardName As String, dblCardAmount As Double, datCardExpire As Date) As Long

Private Methods

GetPaymentID() As Long

Where Used

Order

The SalesTax component computes the amount of sales tax for an order based upon the purchase amount and the state in which the customer resides.

The GetSalesTax method accepts a two character state code and the purchase amount as input. It determines the state's tax rate by reading the StateTaxTable. It computes the sales tax by multiplying the state's tax rate by the purchase amount, and returns the sales tax amount to the caller.

Attributes

ProgID: AWSalesTax.SalesTax
Project file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\SalesTax\AWSalesTax.Vbp
DLL file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Package\AWSalesTax.dll
Transaction: Supports Transactions

Public Methods

GetSalesTax(strState As String, lngSalesAmount As Currency) As Currency

Where Used

Order

The Shipping component is currently a place holder. Its methods do nothing.

Attributes

ProgID: AWShipping.Shipping
Project file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Shipping\AWShipping.vbp
DLL file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Package\AWShipping.dll
Transaction: Requires a Transaction

Public Methods

AddItem(ProductCode As String, ProductName As String, Color As String, Size As String, Quantity As String) As Long Dim ctxObject As ObjectContext

SetHeader(ShipAddress As String) As Long

Ship() As Long

Where Used

Order

The ShoppingBasket component represents a collection of ShoppingBasketItems. It's purpose is to keep track of all the items that the customer has selected for purchase.

The AddItem method creates a new ShoppingBasketItem object and records the appropriate product code, quantity, color, size, and price in the object.

The GetItem method returns the specified ShoppingBasketItem object from the ShoppingBasket collection.

The ItemCount method returns the count of ShoppingBasketItem objects in the ShoppingBasket collection.

The RemoveAllItems method removes all of the ShoppingBasketItem objects from the ShoppingBasket collection.

The RemoveItem method removes the specified ShoppingBasketItem object from the ShoppingBasket collection.

The TotalCost method computes the total cost of all of the items in the Shopping Basket collection.

The private NextItemNumber method assigns a unique number to each ShoppingBasketItem object in the ShoppingBasket collection.

Attributes

ProgID: AWBasket.ShoppingBasket
Project file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Basket\AWBasket.Vbp
DLL file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Package\AWBasket.Dll
Transaction: Does Not Support Transactions

Public Methods

AddItem(ByVal vProductCode, ByVal vUnitPrice As Long, ByVal vDiscount As Long, ByVal vQuantity As Single, ByVal vColor As String, ByVal vSize As String, ByVal vOnsale As Boolean) As Variant

GetItem(ByVal itemNum As Long) As ShoppingBasketItem

ItemCount() As Long

RemoveAllItems()

RemoveItem(ndx As Long)

TotalCost() As Variant

Private Methods

NextItemNumber() As Long

Where Used

check_out.asp
submit_order.asp
Order

Each ShoppingBasketItem object represents an item the customer wishes to purchase. The instance variables in the ShoppingBasketItem object store the quantity, color, size, and price of the item that the customer has selected.

The UpdateItem method is called to change the color, size, or quantity of a previously selected ShoppingBasketItem. This method also updates the extended price if the quantity is changed.

Attributes

ProgID: AWBasket.ShoppingBasketItem
Project file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Basket\AWBasket.Vbp
DLL file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Package\AWBasket.Dll
Transaction: Does Not Support Transactions

Public Methods

UpdateItem(vQuantity, vColor, vSize)

Where Used

check_out.asp
submit_order.asp
Order

The TakeANumber component dispenses unique sequence numbers. Its never dispenses the same number twice, but it does not guarantee that the numbers it dispenses increase monotonically.

The GetAscendingFast method uses the Shared Property Manager to keep track of the Next Number to be dispensed. Each time a number is dispensed the Next Number variable is incremented. The Next Number variable is stored in a Shared Property Group under the control of the Shared Property Manager. The Shared Property Manager provides automatic concurrency control so that two or more clients can safely share the Next Number variable.

Storing the Next Number variable in a Shared Property Group, rather than on disk, makes it much faster to access and update the variable. However, if the Next Number variable were never written to disk, it would be impossible to know what number to dispense after a system failure . To avoid this problem, the Next Number variable is periodically written to disk. We do this each time 100 numbers have been dispensed. If a failure occurs, the system can read the Next Number value from disk, and recover using it. This ensures that the same number is never dispensed twice. A failure may cause up to 100 undispensed number to be skipped. Fortunately, integers are plentiful, failures are rare, and our application can tolerate these occasional gaps.

Attributes

ProgID: AWTakeANumber.TakeANumber
Project file name: TxDemo\ActiveXClient\ProgramFiles\VB5\TakeANumber\AWTakeANumber.vbp
DLL file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Package\AWTake.dll
Transaction: Supports Transactions

Public Methods

GetAscendingFast(strPropGroupIn As String) As Long

GetAscendingSequential(strPropGroupIn As String) As Long

Where Used

Order

This component works in conjunction with TakeANumber. See the description of TakeANumber for details.

Attributes

ProgID: AWTakeANumber.TakeANumberUpdate
Project file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\TakeANumber\AWTakeANumber.vbp
DLL file name: MTxDemo\ActiveXClient\ProgramFiles\VB5\Package\AWTake.dll
Transaction: Requires a new Transaction

Public Methods

Update(lngInc As Long, strPropGroup As String) As Long

Where Used

TakeANumber

Adventure Works uses the following database tables:

® Customers
® Inventory
® OrderDetail
® Orders
® PaymentAuthorization
® Payments
® ProductGroup
® Products
® ProductType
® SalesTaxTable
® TakeANumber

The Adventure Works sample application files are organized in the following directory structure:

```
xDemo
ActiveXClient
     Controls
     Equipment
     Excursions
     Internal
     Multimedia
     ProgramFiles
     Resources
SQLScripts
```

ActiveXClient
This is the central directory for the Adventure Works application.

Controls
Contains the ActiveX controls for the Adventure Works application.

Equipment
Contains the Active Server Pages scripts for the online shopping portion of Adventure Works. This directory contains the scripts that invoke Microsoft Transaction Server components. This directory, and the ProgramFiles directory, will be of the most interest to you if you want to understand Microsoft Transaction Server.

Excursions
Contains the Active Server Pages scripts for the "Base Camp" portion of Adventure Works. This is the part of the web site offering excursions.

Internal
Contains the Active Server Pages scripts for the "Inner Trail" portion of Adventure Works. This is the part of the web site used for internal store management.

Multimedia
Contains the image (.GIF) and sound (.WAV) files for Adventure Works.

ProgramFiles
Contains the Visual Basic project, source, and object files for the online shopping portion of Adventure Works. This directory, and the Equipment directory, will be of the most interest to you if you want to understand Microsoft Transaction Server.

Resources
Contains the Active Server Pages scripts for the "Off The Wall" portion of Adventure Works.

SQLScripts
Contains the Microsoft SQL Server scripts for creating the Adventure Works SQL Server database files. These scripts are invoked by the Adventure Works setup program when you install Adventure Works.

Name	Type	Size
CustomerID	Number (Long)	4
CustomerFirstName	Text	30
CustomerLastName	Text	30
BillingAddressText	Text	255
City	Text	50
StateOrProvince	Text	20
PostalCode	Text	20
Country	Text	50
PhoneNumber	Text	30
EmailAddress	Text	50

Name	Type	Size
ProductCode	Text	10
QuantityOnHand	Number (Long)	4
ReorderQuantity	Number (Long)	4

Name	Type	Size
OrderDetailID	Number (Long)	4
OrderID	Number (Long)	4
ProductCode	Text	10
Quantity	Number (Long)	4
Color	Text	30
Size	Text	30
DiscountedPrice	Number (Double)	8
ExtPrice	Number (Double)	8

Name	Type	Size
OrderID	Number (Long)	4
CustomerID	Number (Long)	4
OrderDate	Date/Time	8
ItemTotal	Number (Double)	8
FreightCharge	Number (Double)	8
SalesTax	Number (Double)	8
OrderTotal	Number (Double)	8

Name	Type	Size
PaymentType	Number (Long)	4
PaymentAuthorize	Yes/No	1

Name	Type	Size
PaymentID	Number (Long)	4
OrderID	Number (Long)	4
PaymentAmount	Currency	8
PaymentDate	Date/Time	8
CreditCardNumber	Text	30
CardHolderName	Text	50
CreditCardExpDate	Date/Time	8
CreditCardAuthorizationNumber	Text	30
PaymentMethod	Number (Long)	4

Name	Type	Size
ProductGroupCode	Text	20
ProductGroupDesc	Text	50

Name	Type	Size
ProductCode	Text	10
ProductType	Text	20
ProductIntroductionDate	Date/Time	8
ProductName	Text	50
ProductDescription	Text	255
ProductSize	Text	5
ProductImageURL	Text	255
UnitPrice	Number (Double)	8
OnSale	Yes/No	1
Discount	Number (Long)	4
ProductImage	Text	255

Name	Type	Size
ProductTypeCode	Text	20
ProductGroupCode	Text	20
ProductTypeDesc	Text	50

Name	Type	Size
State	Text	2
TaxRate	Number (Double)	8

Name	Type	Size
PropertyGroupName	Text	255
NextNumber	Number (Long)	4

Using ADO in an MTS Component

Adventure Works components make extensive use of ADO.

Simple ADO Read

Both the **AuthorizePayment** component's **AuthorizePayment** method and the **SalesTax** component's **GetSalesTax** method do simple ADO reads.

Simple ADO Update

The **Order** component's **UpdateInventory** method does a simple ADO update.

Simple ADO Insert

Both the **Order** component's **AddHeaderToOrder** method and the **Payment** component's **PostPayment** method do simple ADO inserts.

ADO Rowset Read

Both the **Catalog** component's **GetCatalogItemsByProductType** and **GetCatalogItemsByCode** methods, and the **Customer** component's **LookUpCustomerByLastName** and **LookUpCustomerByID** methods do ADO rowset reads.

ADO Rowset Update

The **Customer** component's **UpdateCustomers** method does an ADO rowset update.

ADO Rowset Insert

The **Order** component's **AddItemsToOrder** method does an ADO rowset insert.

Using ADO in an Active Server Page Script

Adventure Works Active Server Page scripts use ADO to read from the database; updates are performed by business objects running in Microsoft Transaction Server.

Rowset Read

equip.asp reads the **Products** table.

Rowset Update

No Active Server Page script updates a rowset.

Rowset Insert

No Active Server Page script inserts a rowset.

Using ADO in a Browser Client Script

Adventure Works Browser Client scripts makes use of ADC.

Rowset Read

select2.asp reads the **ProductType** table.

Rowset Update

CustInfoNew.asp updates the **Customers** table.

Rowset Insert

No Browser Client script inserts a rowset.

Passing A Rowset Between Two MTS Components

No Adventure Works component passes a rowset to another Microsoft Transaction Server component.

Passing a Rowset Between an MTS Component and an Active Server Page Script

Adventure Works passes rowsets between Microsoft Transaction Server components and Active Server Page scripts.

MTS Component Returns Rowset to Active Server Page Script

catalog_type.asp creates a **Catalog** object and calls the **GetCatalogItemsByProductType** method. **GetCatalogItemsByProductType** creates a rowset and returns it to **catalog_type.asp** which builds an HTML page using the contents of the rowset.

Active Server Page Script Passes Rowset to an MTS Component

No Active Server Pages script passes a rowset to a Microsoft Transaction Server component.

Passing a Rowset Between an MTS Component and a Browser Client Script

Adventure Works passes rowsets between Microsoft Transaction Server components and Browser Client Scripts via ADC.

MTS Component Returns Rowset to Browser Client Script

CustInfoNew.asp invokes the **Customers** component's **LookUpCustomerByLastName** method. **LookUpCustomerByLastName** creates a rowset by reading the **Customers** table using ADO. It returns the rowset to **CustInfoNew.asp** which binds the rowset to a client-side ActiveX control where the rowset is displayed.

Browser Client Script Passes Rowset to an MTS Component

After obtaining a rowset as described above, **CustInfoNew.asp** permits the Adventure Works application user to modify the customer information in the rowset. **CustInfoNew.asp** then calls the **Customers** component's **UpdateCustomers** method passing it the modified rowset. **UpdateCustomers** writes the modified rowset back to the **Customers** database table.

Invoking an MTS Component from Another MTS Component

Adventure Works includes many examples of one Microsoft Transaction Server component invoking another. For example:

The **Order** component's **GetSalesTax** method invokes the **SalesTax** component's **GetSalesTax** method.

The **Payment** component's **PostPayment** method invokes the **AuthorizePayment** component's **AuthorizePayment** method.

Invoking an MTS Component from an Active Server Page Script

Adventure Works includes two examples of a Microsoft Transaction Server component being invoked from an Active Server Pages script.

catalog_type.asp invokes the **Catalog** component's **GetCatalogItemsByProductType** method.

submit_order.asp invokes the **Order** component's **Submit** method.

Invoking an MTS Component from a Browser Over HTTP

Adventure Works includes one example of a Microsoft Transaction Server component being invoked from a browser over HTTP.

CustInfoNew.asp invokes the **Customers** component's **LookUpCustomerByLastName** and **UpdateCustomers** methods.

For more information regarding invoking Microsoft Transaction Server components from a browser over HTTP, refer to the Advanced Data Space (ADS) feature in the *Advanced Data Connector (ADC)* documentation. Click here to connect to the ADC Web page to review this documentation.
{ewc mvimg, mvimage,!intjump.bmp}

Note that there are currently restrictions regarding the type of information your component can return to the caller when invoked through ADS. At present, the component cannot return any output parameters to the caller. The component may only return a single variant result and this variant result cannot be an array, though it may be a rowset. These restrictions may be relaxed in future releases.

Also note that the registry on the server system containing the called component must be updated to allow the component to be invoked remotely through ADS. Under the:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters
    \ADCLaunch
```

key you must add the ProgID of the called component.

Invoking an MTS Component from a VB Program Running on Windows 95 or Windows NT Over DCOM

For instructions on invoking an MTS Component from a Visual Basic Program Running on Windows 95 or Windows NT Over DCOM see Running Bank on a Remote Client.

Adventure Works includes one example of downloading and using an ActiveX control.

select2.asp produces HTML and VB Script that both downloads an ADC Client control and manages this control.

Adventure Works includes one example of using the Shared Property Manager.

The **TakeANumber** component's **GetAscendingFast** method uses the Shared Property Manager to dispense sequence numbers.

You can go directly to a demonstration by clicking an icon or topic title below.

{ewc MVIMG_MVIMAGE_idemoclip.bmp}

Using the ScriptActive Plug-In

This section describes various Microsoft and third-party developer resources that may be of use to Web site developers.

® Microsoft Programs

Details how to become a Microsoft Certified Professional and Microsoft Solution Provider, and tells how to participate in Microsoft logo programs.

® Microsoft Technical Support

Points you to the *Mastering Web Site Development* Help file for information about online, CD-ROM-, Fax-, and phone-based support for Microsoft products.

® Microsoft Technical Training and Education

Describes the training options available through Microsoft — face-to-face, self-paced, and classes offered through the Microsoft Online Institute.

® Lists and References

Provides a listing of user groups and mailing lists that might be of interest to Web site developers.

® Publishers

Provides information about Microsoft Press and third-party publishers who create books and periodicals that include Web site development topics.

® World Wide Web Sites

Provides pointers to many Web sites that might be of interest to Web site developers.

® Tools

Provides information and instructions for installing several tools included on the *Mastering Web Site Development* CD-ROM, in the Media\Tools directory. Most of these tools can be automatically installed or executed by clicking the appropriate Launch Executable icon.

Mastering Web Site Development Using Visual InterDev provides two sample applications in the SampApps folder on the CD-ROM:

® Adventure Works

The Adventure Works sample application illustrates how an Internet business application can be created using Microsoft's Active Desktop and Active Server technologies.

® Up and Running with Microsoft Visual InterDev Tutorial

This tutorial allows you to test-drive Visual InterDev and explore some of its key features by providing step-by-step instructions.

The Sample Bank application on the Microsoft Transaction Server illustrates how easy it is to build a Microsoft Transaction Server application. For instructions on installing and running the Bank application, see Microsoft Transaction Server Sample Bank.

There are also numerous downloadable sample applications on the Microsoft Visual InterDev Sample Applications Web page. Here is a list of the other sample applications (and their size) currently available on the Web site. Click here to connect to the Web site for downloading instructions.
{ewc mvimg, mvimage,!intjump.bmp}

® **401(k)** (395 KB)

An intranet application that shows how a Fortune 1000 company can manage benefits using applications built in Visual InterDev. Employees can choose how much money to deduct from their paychecks and how they want to invest that money. They can also change investment options and view their portfolios. This sample is also available on the Visual InterDev installation CD.

® **Dos Perros Chile Company** (1536 KB)

This sample Web online shopping application demonstrates server-side scripting and maintaining state. The sample includes an overview of Visual InterDev, "Building Visual InterDev Applications." This sample is also available on the Visual InterDev installation CD.

® **Friendship Insurance** (2184 KB)

The Friendship Insurance sample is comprised of two related applications, one for the Internet and one for an intranet, that provide a variety of advanced concepts for Web development. The Internet application provides Friendship customers with the ability to login to Friendship Online, and view and modify their insurance policies. It comes complete with authentication logic, database-bound HTML forms, and even a cool Java applet that talks directly to the database! The intranet application allows Friendship agents to review and finalize Internet-submitted policy submissions. It comes complete with authentication logic, a Microsoft Visual Basic version 5.0 Active Server Component, database connectivity, and Active Document Support.

® **Parameterized Stored Procedure** (46 KB)

This sample shows how to use Visual InterDev design-time controls to easily implement parameterized stored procedures in a Web application. The sample demonstrates how to run a parameterized stored procedure via an Active Server Page to return database records from a SQL Server database.

® **Parameterized Query: SQL version** (45 KB)
Parameterized Query: Microsoft Access version (60 KB)

This sample shows how to use Visual InterDev design-time controls and the Visual InterDev Query Designer to easily implement parameterized queries in a Web application. The sample demonstrates how to run a parameterized query via an Active Server Page to return database records from a database.

® **Parameterized Data Form** (58 KB)

This sample shows the use of parameterized queries in conjunction with the Data Form Wizard. The HTML data form is driven by a user selecting from a list of records displayed on another page. This sample can help you set up a variety of scenarios such as Master-detail drill-down pages between different queries, different tables, and potentially even different databases.

Mastering Web Site Development contains a number of code samples.

When you see this icon {ewc MVIMG, MVIMAGE,!code.bmp}, click it to view the sample code.

This course includes a number of self-check questions at the end of each chapter. You can use these multiple-choice questions to test your understanding of the information covered in the chapter.

To see whether an answer is correct or incorrect, click this icon {ewc MVIMG, MVIMAGE,!answer.bmp}.

Each answer contains a jump to the associated chapter, so you can easily review the content.

You can review the self-check questions at the end of each chapter, or you can go to them directly by clicking a chapter title below.