Deploying Solutions with .NET Enterprise Servers

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Deploying Solutions with .NET Enterprise Servers

Mike Young
Curtis W. Young
Gearhead Press™

Wiley Publishing, Inc.

Publisher: Joe Wikert

Editor: Ben Ryan

Editorial Manager: Kathryn A. Malm

Developmental Editor: J. W. Olsen

Managing Editor: Micheline Frederick

Text Design & Composition: Wiley Composition Services

This book is printed on acid–free paper. ♻

For general information on our other products and services please contact our Customer Care Department within the United States at (800) 762–2974, outside the United States at (317) 572–3993 or fax (317) 572–4002.

A Note from Gearhead Press

Gearhead Press is dedicated to publishing technical books for experienced Information Technology professionals—network engineers, developers, system administrators, and others—who need to update their skills, learn how to use technology more effectively, or simply want a quality reference to the latest technology. Gearhead Press emerged from my experience with professional trainers of engineers and developers: people who truly understand first–hand the needs of working professionals. Gearhead Press authors are the crème de la crème of industry trainers, working at the companies that define the technology revolution. For this reason, Gearhead Press authors are regularly in the trenches with the developers and engineers that have changed the world through innovative products. Drawing from this experience in IT training, our books deliver superior technical content with a unique perspective that is based on real–world experience.

Now, as an imprint of Wiley Publishing, Inc., Gearhead Press will continue to bring you, the reader, the level of quality that Wiley has delivered consistently for nearly 200 years.

Thank you.

Donis Marshall
Founder, Gearhead Press
Consulting Editor, Wiley Publishing, Inc.

I dedicate this book to my wife. She has been patient with me and continued to be diligent in raising our children as I invested the necessary efforts and time in this endeavor.

—Mike Young

About the Authors

Mike Young is the cofounder of SofTouch Inc. Mike has spent the last several years teaching, consulting, and developing training materials for Microsoft server products. Recently, Mike has

focused his energies on the .NET Enterprise Server suite. Within that suite, Mike has spent the majority of his time supporting and consulting about Microsoft SQL Server. He has a background in database administration, Web server administration, and .NET Server integration. Mike is particularly concerned that his clients meet their expectations for the product. His primary areas of expertise are Data Transformation Services, Analysis Server, and all areas related to security.

Curtis Young is the other cofounder of SofTouch Inc. Curtis has a deep love of training and education. His background is on the programming side. He has taught and consulted regarding Visual Basic, Visual C++, and Java. Over the last several months Curtis has changed the focus of his consulting to the new .NET products. He has invested a significant amount of time in designing and developing applications that use Visual Basic .NET and C# as development tools and SQL Server as the back−end database. He receives the most satisfaction from providing systems solutions to business obstacles.

Acknowledgments

Introduction

Over the last several years we have spent the majority of our time supporting and teaching BackOffice Products. While we are supporters of the Microsoft application servers, we also have experienced the headaches associated with enterprise development and application integration. It seems as though the application servers were written to run individually rather than in association with each other.

The .NET Enterprise Servers provide an answer for many of these headaches. It is exciting to see each of the servers grow as Microsoft recognizes the need to integrate servers for full–service business solutions.

We have a desire to help the reader see the vision of the .NET Enterprise Servers. This book has been written from the perspective of the solutions that developers may need to create. The .NET Enterprise Servers are the tool, not the solution. In reality, the solution may require multiple applications that work together to fulfill the developer's business need. This book is designed to help readers recognize how they can better fulfill their business needs with the Microsoft .NET Enterprise Servers.

Overview of the Book and Technology

The .NET Enterprise Servers are a suite of products that provide an application server framework for development needs. Microsoft previously supported the BackOffice suite of products. The .NET Enterprise Servers are the upgrade to BackOffice. Many of the products in the suite have been enhanced. Several of the products, such as Application Center 2000 and BizTalk Server, are new to the suite of server products.

The purpose of the .NET Enterprise Servers is to provide a development and deployment platform for business solutions. Microsoft released this line of products to meet developers' integration requirements. For years many organizations have struggled with the process of integrating different server platforms. In response to this concern, Microsoft has focused on creating a platform infrastructure that is "enterprise–development–friendly." The .NET Enterprise Servers not only integrate well with each other but also can be used to integrate with other vendors' server products.

This book is unique in the breadth of topics covered. Each of the .NET Enterprise Servers is described in some depth. The book assumes that the reader has a basic understanding of the products and some of their features. The book focuses on how each product enhances or uses the .NET services and how it will help solve business process requirements.

This book is designed to help readers see Microsoft's vision for the .NET Enterprise Servers. Examples illustrate how each of these servers can be used to help create business solutions.

How This Book Is Organized

This book's chapters are divided into three distinct parts:

- **Part One**: **.NET Overview.** This part includes introductions to the .NET Framework, the .NET Enterprise Servers, and .NET business solutions. It also introduces two of the core .NET services: Extensible Markup Language (XML) and .NET components. These foundation chapters lay the groundwork for the business solutions and products that are

described in the rest of the book.

- **Part Two: .NET Enterprise Servers.** In this part a chapter provides a description of each of the .NET Enterprise Servers. Each of the products is presented from the standpoint of how it interacts with .NET services and how it can be used to enhance business solutions. Within each chapter the reader is first introduced to the product. Then the product is presented in relation to the .NET services. It is important to understand how each of the products uses or enhances the .NET services, such as XML, .NET components, and Web services. Finally, each chapter presents the business solutions that can be accomplished with the product.
- **Part Three: Enterprise Solutions with .NET Servers.** The final part of this book describes some business solutions that include several of the products. Within this final part the reader is introduced to some of the common implementations of the .NET Enterprise Servers. Some of these technologies are clustering, Web farms, data warehousing, and building Web services.

This book consists of 18 chapters and 2 appendixes:

- **Chapter 1: Introduction to the .NET Framework.** The book begins by introducing the .NET architecture, the first environment constructed from the ground up for Internet development. The reader will first examine the three fundamental layers of the .NET Framework: the common language runtime, the framework's base classes, and its user and program interfaces. The reader also is exposed to the role they play for a .NET developer. The reader will find an overview of the changes to the Visual Basic and C++ languages and be introduced to Microsoft's newest .NET language, C# (pronounced "C sharp").
- **Chapter 2: .NET Business Solutions.** This chapter introduces the various types of business solutions that are possible with the .NET Enterprise Servers. The chapter is broken down into three main types of solutions: enterprise application integration (EAI) solutions, business–to–business (B2B) solutions, and business–to–consumer (B2C) solutions. This chapter provides a foundation for many of the other chapters in the book. Each product that is introduced is presented from the perspective of how it enhances or makes possible each of these .NET business solutions.
- **Chapter 3: Role of .NET Enterprise Servers.** This chapter provides an overview of each of the products that make up the .NET Enterprise Server suite. Each of the server products is described in terms of its design goals and then in regard to the features it provides to meet those goals.
- **Chapter 4: .NET Services for the Development Environment.** This chapter deepens the reader's understanding of the three layers of the .NET Framework and the services they provide to a .NET developer. The reader gains a perspective on the true multilanguage development made possible by the common language runtime. The reader is exposed to other common language runtime services as well as the breadth of functionality embedded in the .NET Framework's base classes. The chapter concludes with an introduction to ASP.NET and Web forms, both of which are elements of the user and program interfaces layer of the .NET Framework.
- **Chapter 5: XML Integration with .NET Servers.** XML is the communication medium for all .NET technologies and servers and thus is the foundation of the .NET Framework. This chapter introduces this "markup" language and its focus on data structures and transformation. The reader learns about the fundamental technologies underlying and supporting XML and their role in the .NET platform.
- **Chapter 6: Building .NET Components.** This chapter examines the structure of .NET components and their new functionality in distributed application development. The chapter examines the nature and role of assemblies in .NET development and deployment. The reader also will see how .NET coexists with components developed under Microsoft's prior component technology, Component Object Model (COM). Additionally, the chapter

examines how .NET consumes and requires Component Services (COM) to fulfill its new role in distributed application development.

- **Chapter 7: Exchange Server.** Microsoft's email server has been upgraded for the 2000 version. This chapter identifies the new features of Exchange Server 2000 that enhance enterprise development. Readers are introduced to the Web storage system and the Installable File System, both of which enhance the opportunity to access the data that have been stored in Exchange Server.

- **Chapter 8: SQL Server 2000.** SQL Server 2000 acts as the relational database management system for most of the solutions presented in this book. This chapter introduces the features and services that SQL Server 2000 provides for .NET. Specifically, a large proportion of the chapter is dedicated to XML integration with SQL Server and building Web services from SQL Server stored procedures.

- **Chapter 9: Application Center 2000.** Application Center 2000 is a powerful new product that is beneficial to developers and administrators. Application Center 2000 can load−balance Web and application servers. The requests of the servers are alternated between the various servers in the server group (cluster). Additionally, Application Center 2000 can synchronize the servers in the cluster. Updates can be performed to one of the machines, and then the others are updated automatically. Application Center 2000 enhances the availability and performance of Web and application servers.

- **Chapter 10: Commerce Server.** E−commerce sites have been a buzzword in this industry for the last couple of years. Commerce Server assists with e−commerce solutions. It can be used to extend the normal security of Internet Information Server, make a product catalogue available to Web users, and browse through a data warehouse of the activity against a site. Commerce Server is a key component in maximizing B2C solutions.

- **Chapter 11: Content Management Server.** Content Management Server can reduce the amount of time it takes to build and deploy a Web site dramatically. Through templates and dynamic content delivery, a company can customize a site to its organizational requirements. After reading this chapter, the reader will be able to determine whether Content Management Server will bring a return on the investment.

- **Chapter 12: BizTalk Server.** BizTalk Server is a powerful component for EAI and B2B solutions. BizTalk Server can use XML to transform and map data between multiple data sources. The resulting applications can be organizational applications or those of vendors or trade partners.

- **Chapter 13: SharePoint Portal Server.** SharePoint Portal Server integrates nicely with users. This server is focused on enhancing the way in which users can access and share data. With SharePoint Portal Server one can extend the functionality of products such as Exchange Server and Content Management Server to the user desktop. After studying this chapter, the reader will have a clear picture of how this product will enhance users' day−to−day work experience.

- **Chapter 14: Internet Security and Acceleration Server.** Internet Security and Acceleration (ISA) Server is Microsoft's upgrade to Proxy Server 2.0. ISA Server is Microsoft's enterprise−class firewall. This chapter focuses on the features related to the firewall and the options for getting requests through the firewall. In many cases the firewall appears to be a deterrent to effective application development. This chapter outlines the steps one should take to use the firewall effectively without losing the functionality of applications.

- **Chapter 15: Web Services with .NET.** Through a Web service it is possible to make a process available to the Internet or an intranet. Other application developers then can use the process within their applications. An example of this technology is stock quotes. A Web service could exist with the current stock information, and then any Web developer could include the service from his or her application. This chapter discusses the role of Web services, how to create Web services, and how to use Web services.

- **Chapter 16: Building Enterprise Web Farms.** Enterprise Web farms should be available,

secure, and responsive. This chapter outlines the layers of a Web farm and explains how to meet each of the design goals at each layer of the implementation. With a Web farm a portion of the application should be outside the firewall, some of the application should be behind the firewall, and the data should be stored in a database management system. This chapter addresses each of these layers.

- **Chapter 17: Data Warehousing.** Data warehousing provides a level of analysis that is desired by many organizations. This chapter introduces the Analysis Services of SQL Server 2000 as a data warehousing solution. The chapter then identifies options for making the solution available over a Web medium.
- **Chapter 18: Clustering .NET Servers.** Clustering boosts the stability and availability of data. Data are stored in a storage area network (SAN), and the user interfaces with servers (nodes) that provide access to the data. If one of the nodes goes down, another picks up the requests, and end users never know the difference. Clustering is the ultimate fault–tolerance solution.
- **Appendix A: Visual Basic .NET Reference Guide.**
- **Appendix B: The Role of Visual Studio .NET.** This appendix focuses primarily on the details of the products that are included with each of the versions of Visual Studio .NET. After reading the book, the reader may wonder how he or she can obtain each of the products and which products are available with each version of Visual Studio .NET. After reading this appendix, the reader should feel comfortable deciding which version of Visual Studio .NET best meets his or her current requirements.

Who Should Read This Book

This book was written with the enterprise developer in mind. Many of the examples and chapters are presented from the perspective of the application developer. A developer will find that this book presents many concepts that he or she may want to implement. In many cases, the examples and services presented focus on the design of the implementation. The reader will come away from this book with a better understanding of how each of the .NET Enterprise Servers can be used to facilitate enterprise development.

Many of us have the tendency to limit ourselves to the products and development tools we know. This book can help the reader extend his or her horizons to see what can be possible through the use of the .NET technologies and the .NET Enterprise Servers.

Additionally, this book can be useful for information technology (IT) decision makers. It can help the reader determine the products that he or she may want to purchase and shows how those products would enhance the reader's environment. Each chapter provides examples and an overview of the product presented. Decision makers will find this book valuable in purchasing new products and determining the appropriate use of the products in the organization.

This book is not written from the perspective of a network administrator, but an administrator can take advantage of the book. The reader can get a better high–level understanding of the types of applications that can be created by his or her developers. This knowledge can be beneficial when it is time to evaluate new products or troubleshoot problems with existing solutions that use the .NET Enterprise Servers.

Tools That Will Be Needed

This book does not list a lot of required underlying tools, though most of the chapters include step−by−step procedures and examples. This book is unique in the breadth of technical information supplied. If the reader does not have one or more of the .NET Enterprise Server products, he or she will not be able to complete the specific step−by−step procedures, but that will not detract from the experience of reading the book.

The following products are used in the chapters and appendixes, and one can benefit from access to them while reading this book. If the reader does not have a specific product installed, he or she should read the chapter about that product and determine whether it is needed to fully understand its role in the .NET Enterprise Server suite:

- .NET Advanced Server
- Visual Studio .NET (Visual Basic .NET as the development tool)
- Exchange Server 2000
- SQL Server 2000
- Application Center 2000
- Commerce Server 2002
- Content Management Server
- BizTalk Server
- SharePoint Portal Server
- Internet Security and Acceleration Server

Summary

The authors believe that the reader will find that this book is a refreshing introduction to the business solutions that are possible through the use of the .NET Enterprise Servers. The book focuses on integration and full−service solutions to business requirements. After finishing this book, the reader will come away with new ideas for the use of his or her current server products and a forward−thinking approach to the new products that are included with the suite.

Part I: .NET Overview

Chapter List

Chapter 1: Introduction to the .NET Framework

Highlights

Although the goal of this book is to introduce deployable solutions with the .NET Enterprise Servers, it is necessary to begin by examining the architecture of the .NET Framework. The .NET Enterprise Servers that will be introduced in later chapters will be discussed as tools that can be exploited by .NET developers. As a new, integrated software development environment, the .NET Framework allows the developer to combine the front-end development environment easily with the .NET Enterprise Servers to facilitate the development of scalable, Internet-ready applications. The framework is the foundation for the .NET programming environment and should be researched thoroughly by all developers who need to create new Windows applications.

The focus of this book is on building the .NET Framework by demonstrating the power that can be derived from the new .NET Enterprise Server architecture rather than on the technical aspects of the framework's components. For more information on the specifics of the .NET Framework, the reader can access Microsoft's Web site at www.microsoft.com/net.

This chapter first provides an overview of the .NET Framework. The framework is defined in a three-layer model. The model includes the common language runtime, the .NET Framework base classes, and the user and program interfaces. The section on user and program interfaces specifically defines the purpose of both Windows Forms and Web Forms.

After the overview of the .NET Framework, the chapter highlights some of the changes to the Visual Basic and Visual C++ programming languages. The chapter then introduces C# .NET and identifies its role in the new development framework. A short section discusses the changes to Active Server Pages (ASP). Finally, the chapter outlines the benefits of the .NET Framework to the developer.

By the end of this chapter the reader will have a deeper understanding of the role of the .NET Framework and the purpose for the various programming languages. The reader also will learn why the .NET Framework will become the solution of choice for Windows developers.

.NET Framework Overview

Microsoft's new software development platform, .NET Framework, is the first Microsoft development environment designed from the ground up for Internet development. Although .NET is not meant to be used exclusively for Internet development, its innovations were driven by the limitations of current Internet development tools and technology.

The basis of this new development platform consists of three primary components or layers: the common language runtime, the .NET Framework base classes, and the user and program interfaces, as demonstrated in Figure 1.1.

Components of .NET Framework
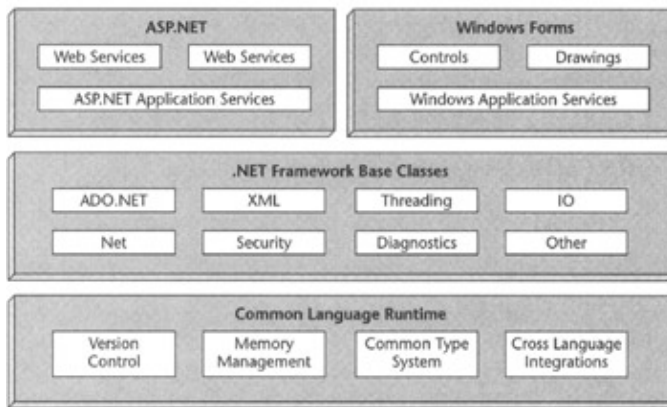
Figure 1.1: The .NET Framework has three layers: common language runtime, the .NET Framework base classes, and the user and program interfaces (ASP.NET and Windows Forms).

The foundation of the .NET Framework is the common language runtime. Its principal purpose is to load, execute, and manage code that has been compiled to Microsoft's new intermediate byte–code format called Intermediate Language (IL). Several languages, notably Microsoft's Visual Basic .NET and C# .NET (pronounced "C sharp"), have compilers supporting this format, and many more are currently in development. It is important to note that the IL code is not interpreted. The common language runtime uses just–in–time compilers to compile the IL code to native binary code before execution.

Other significant features of the common language runtime include the following:

- Version control
- Memory management
- Cross–language integration
- Common data type system

The next component or layer of the .NET Framework is the .NET Framework base classes. The purpose of this layer is to provide the services and object models for data access and manipulation, data streams (input/output [I/O]), security, thread management, and more. In many respects the Windows API (Application Programming Interface) has been abstracted into these base classes. These libraries are an object–oriented collection of classes that can be reused and enhanced to enable rapid application development. The classes support the creation of applications that range from ASP.NET Web pages and Web Services to traditional Windows and command line applications.

The .NET Framework also provides several runtime hosts, which are unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, creating a software environment that can exploit both managed and unmanaged features. The .NET Framework software developer's kit (SDK) not only provides several runtime hosts but also supports the development of third–party runtime hosts.

For example, ASP.NET hosts the runtime to provide a scalable, server–side environment for managed code. ASP.NET works directly with the runtime to enable .aspx pages and Web services, both of which are discussed later in this section.

The final layer of the .NET Framework consists of the user and program Interfaces. The two components of this layer are ASP.NET Application Services and Windows Application Services. The cornerstone of the ASP.NET Application Services is, of course, ASP.NET, which in turn supports the new Web services and Web Forms technologies that are discussed later. The Windows

Application Services component supports traditional Windows programming applications through Windows Forms.

The following sections describe the main components of the .NET Framework in more detail.

Features of the Common Language Runtime

The common language runtime provides an execution environment that manages the set of code targeted to the .NET Framework. Code management can include memory management, thread management, security management, code verification, and compilation of the code. All managed components must first be assigned a level of trust. The level or degree of trust can vary on the basis of the following origins:

- Local computer
- Internet connection
- Enterprise local area network (LAN) connection

After the component is assigned an appropriate level of trust, the managed component either can or cannot perform functions from within the application. In fact, based on the trust levels, a managed component may act differently within the same active application. The degree of trust can be used to limit registry access, file access, or network functionality. The common language runtime enforces security in a way that enables a client to run executables without endangering or risking inappropriate access to sensitive local resources or devices. For example, a user could double–click an executable in an email, and although it may play a video or audio stream, access to personal data on the computer is not at risk. This enables an Internet application to be as feature–rich as normal desktop applications.

The common language runtime also uses a common type system (CTS) to enforce code robustness and language interoperability. Various Microsoft and other third–party compilers can generate managed code that complies with the CTS. The purpose of the CTS is to enforce a strict data type and code verification standard. Through the CTS, managed code can include managed classes, types, and other objects while enforcing type safety and conformity to the infrastructure.

Another primary goal of the common language runtime is increased programmer efficiency. With all the .NET Framework languages complying with the common language runtime, the programmer can choose a language of preference. Each language can take full advantage of the common language runtime, the class library, and the components. This will make future programming available in whichever language is best suited to the developer and application requirements while maintaining a high level of language interoperability and independence.

The built–in automatic memory management also enhances application and infrastructure stability. This new feature—the garbage collector—eliminates some of the most common application errors involving memory leaks and invalid memory references. It manages all references to objects and releases them when they are no longer in use. In a nutshell, the automatic memory management helps the application clean up after itself.

The common language runtime also should increase application performance. This may be accomplished in two ways: just–in–time (JIT) compliers and server–side applications. The JIT compilers are used to compile all managed code to native machine binary code at execution. Server–side applications can house application logic and business rules on .NET Enterprise Servers such as Internet Information Server (IIS) and SQL Server. This allows the consolidation of application logic and business rules in a more maintainable environment and permits execution

closer to dependent resources. For many applications this will result in not only reduced maintenance requirements but increased responsiveness.

.NET Framework Class Library

The .NET Framework class library is a collection of reusable classes, or types, that tightly integrate with the common language runtime. .NET applications benefit from using and extending or inheriting the functionality from the classes and types available in the class library. The class library is very hierarchical and well organized, as shown in Figure 1.2. It starts with the most generic classes and continues to build down to classes with very specific and precise functionality. Although this library is extensive, its organization makes it easy to learn and use. In an age of ever-growing technology it is refreshing to see a new technology and a new architecture that promise a reduced learning curve. This model also makes it easy for third-party components to be integrated easily with the existing class library.

Unified Programming Model
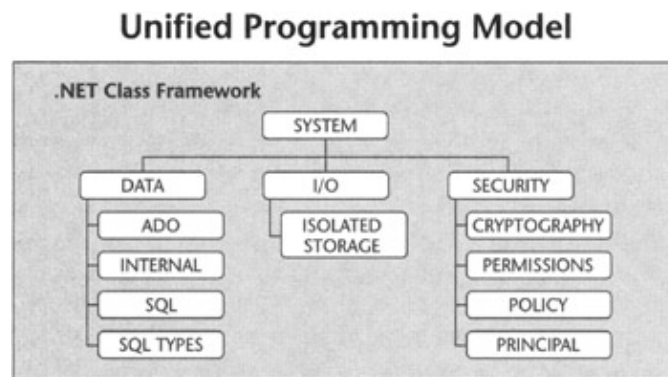
Figure 1.2: The .NET Framework class library is hierarchical in nature and goes from the more generic to the more specific.

As expected in an object-oriented class library, the .NET Framework classes enable developers to accomplish rapidly a wide range of common programming tasks, including things such as string management, file access, and database connectivity. Also, several classes facilitate highly specialized and custom development environments. These classes make the application development environment very flexible. The following types of applications are readily supported through the .NET Framework:

- ASP.NET applications
- Console applications
- Scripted applications
- Windows applications (Windows Forms)
- Web services

For example, the Windows Forms classes are used to create Windows graphical user interface (GUI) applications, which often are referred to as standalone applications. This is facilitated through a series of reusable graphical interface classes. Alternatively, in developing a Web-based application, the HTML and Web Forms classes facilitate its rapid development. Either way the underlying framework provides the flexibility for feature-rich applications regardless of the choice of application environment.

Client Application Development

Client applications represent the most common application environment in use today. These are the applications that one would invoke by opening some type of form and initiating an action. Examples

of client applications range from word processing applications to a customized accounting package. One typically begins the application by opening a form from a desktop icon or the Windows Start menu. Client applications typically use the standard Windows features, such as dialogue boxes, menu items, and buttons. In most cases the files and executables for the application are stored and run locally and are interacted with by using local peripherals.

The traditional Windows programming environment has been replaced in .NET by the Windows Forms control. The managed Windows Forms control allows the application to be deployed over the Internet, and the user can view the application as a Web page.

In the past developers created such applications by using C or C++ in conjunction with the Microsoft Foundation Classes (MFC) or with a rapid application development (RAD) environment such as Microsoft Visual Basic. The .NET Framework incorporates aspects of the earlier products into a single, consistent development environment. The goal of the single environment is to simplify the development of client applications.

This simplicity can be further enhanced by incorporating the security and deployment features of the common language runtime. Because feature-rich applications can be deployed from the Web, many applications that once were installed on a user's system can now be Web-based. The application can define code-access security limits, which restrict what the application can do on an individual's machine. This gives the developer security features even though the application is running in the machine language. In fact, a single application can incorporate objects from multiple Web servers. Each object and Web server may have security rights different from those of another server. All security rights are evaluated even though the objects are used within a single application.

Windows Forms applications still have some advantages over Web-based applications. Windows Forms applications have a level of trust that is already assumed. This allows binary and natively executing code to interact with some of the resources on the local machine. This is used to perform the necessary GUI elements of the application.

Server Application Development

Web services are server-side application components that can be distributed easily in a similar fashion to Web sites. Web services components are not targeted for a specific browser and have no user interface. They are simply reusable software components designed to be used by other applications. Web-based applications, traditional applications, and even other Web services can use the Web services. The Web services are moving application development toward the Internet. All application development can now take advantage of the highly distributed world that previously was available only with Web-based applications.

ASP.NET is the environment that enables developers to use the .NET Framework to target Web services applications. Both Web Forms and Web services use IIS as the publishing mechanism for applications, and both have a collection of supporting classes in the .NET Framework.

If you are familiar with earlier versions of ASP technology, you will enjoy the new features of Web Forms. The most noticeable new feature is the ability to develop one's Web Forms in any language that supports the .NET Framework. You can use Visual Basic .NET to create your Web Forms if that is the language you prefer. You are not required to learn multiple languages to create different types of applications. Web Forms also can execute in the native machine language and take advantage of the common language runtime in the same manner as all other managed applications. This allows the Web Forms to be compiled and executed at a much faster rate. Users will notice that the ASP.NET pages are faster, more functional, and easier to create.

The .NET Framework also provides a collection of classes and tools to aid in the development and consumption of Web services applications. Web services are built on standards such as SOAP (a remote procedure–call protocol), XML (an extensible data format), and WSDL (Web Service Description Language). The .NET Framework conforms to these standards to promote interoperability with non–Microsoft solutions. Although many of the chapters in this book define the role of these protocols and show how they allow the .NET Framework to interact with .NET Enterprise Servers, readers can reference Chapter 15, "Web Services with .NET," for more information on the role of each of these standards.

The .NET Framework provides a set of classes that conform to the underlying standards of SOAP, WSDL, and XML. Consequently, as the user develops and publishes Web services, he or she just focuses on the underlying logic and function of the service, not on infrastructure communication issues.

Web Forms and Web services are compiled and executed in native machine language and take advantage of the scalable architecture of IIS. While these technologies are dependent on IIS, their tight integration makes initial configuration and ongoing maintenance easy.

Windows Forms

As the technologies shift more toward the Web environment, it may appear that the .NET Framework is discouraging traditional Windows application development. This is far from accurate. The Windows Forms environment simplifies the development of applications and can be used to create feature–rich user interface Windows applications.

Windows Forms is a new forms package that can be used to develop and deploy Windows applications through the .NET Framework. Windows Forms is included as part of the new Microsoft Visual Studio package. It uses the .NET platform and leverages the new technologies, including a common application framework, managed execution environment, integrated security, and object–oriented design principles. In addition, Windows Forms offers full support for Web services and can connect easily to data stores by using a .NET data model called ADO.NET. The shared environment of Visual Studio allows a developer to create Windows Forms applications by using any of the languages that support the .NET platform. These languages include Visual Basic .NET, C++, and C#.

Windows Forms in the .NET Framework supports the development of Windows user interface applications. While the design is similar to that of Web Forms, the classes and implementation are significantly different. The development environment, however, is such that an application developer will feel equally comfortable writing an application using Windows Forms or Web Forms. Both environments have similar classes and some events in common, and the classes that are different are at least consistent in function and purpose. Although this book is not necessarily about creating Windows Forms or Web Forms, it is necessary to understand the foundation to be able to build on it with the .NET Enterprise Servers.

The primary goals of Windows Forms are ease of use and reusability. All Windows Forms are based on a common language runtime feature called delegates. Delegates are basically pointers. Each event can be given a delegate handler so that the user is not required to define the handler through another means (override, event map, and interface for all events on the class).

The process of creating a Windows Forms project is also easy. Creating a Windows Forms project with Visual Studio .NET creates only one project file that is compiled: Form1.cs. There are no

header files, no interface definition files, no bootstrap application files, no resource files, and no library files. The goal is to keep the process minimal and easy to coordinate. All the information needed for the project is contained in the code for the form.

Because Windows Forms are built on the common language runtime, developers can choose any one of the many languages that are now targeting the common language runtime to build Windows applications. Developers can now write Windows Forms applications (or Web Forms applications or data applications) in a variety of languages that range from C# to COBOL, Eiffel, and Perl, among others. This decreases the learning curve by allowing developers to stick with their language of choice.

Windows Forms takes full advantage of GDI+, Microsoft's next–generation 2–D graphics system. The graphics programming model in Windows Forms is fully object–oriented, and the assorted pens, brushes, images, and other graphics objects follow the same ease–of–use guidelines as the rest of the .NET Framework. Developers can now include great new drawing features such as alpha blending, color gradients, textures, anti–aliasing, and vector image formats. When this feature is coupled with the Windows 2000 operating system's layered and transparent windows features, developers can create richer, more graphical Win32 applications with much less effort.

Web Forms

Web Forms are the forms that are engine–supplied with ASP.NET. Their purpose is to provide Web browser–based user interfaces. This technology has been enhanced to provide the next generation of development by adding features such as drag–and–drop development.

Web Forms consist of two parts: a template and a component. The template contains the layout information for the user interface elements. The component contains all the logic that is linked to the user interface. Web Forms take advantage of controls to make all the coordination required appear easy.

Controls run on the server and make themselves known to the client. This allows the application interface to have many of the characteristics of a normal Win32 application. Controls are reusable user interface elements that are used to construct the form.

New Programming Languages

Visual Studio .NET is Microsoft's .NET Framework development environment. Unlike earlier releases, this integrated development environment (IDE) is designed to support all programming languages that compile to Microsoft's Intermediate Language (MSIL), the "managed code" that actually is executed by the common language runtime. To meet this requirement, the Visual Studio .NET releases of Visual Basic and C++ underwent significant changes. In addition, Microsoft introduced C#, a new .NET programming language. To fully exploit the new capabilities of the .NET Framework, the changes to Visual Basic were particularly significant. While the .NET release is not compatible with prior versions of Visual Basic, Microsoft has incorporated a migration tool to ease the conversion of existing applications.

However, because the architecture of .NET is so different from the previous application development architecture, many applications may not port easily from the older platform to .NET. Consequently, many Visual Basic Version 6 applications will either be around for some time or

undergo complete rewrites for the new .NET Framework.

The following sections introduce and reference the most popular new features of Visual Basic .NET, C#, and ASP.NET and discuss a couple of considerations for C++.

Visual Basic .NET

This release of Visual Basic represents the most significant architectural change since the initial release of Visual Basic. Because it now shares its development environment with other .NET languages, Visual Basic. NET can draw on the functionality of the .NET Framework base classes. Arguably the most significant advantage this gives Visual Basic developers is its tight integration with Web–based development. Visual Basic has long been the dominant programming language for Windows–based applications, but it always lacked an elegant approach to developing Internet applications. This release changes that situation. In addition to integration with the .NET Framework, Visual Basic .NET incorporates several new features, listed below, which expand the scalability and robustness of Visual Basic application development.

- Object–oriented development has been expanded to be fully supported in Visual Basic .NET. This includes full inheritance, parameterized constructors, and control over property and method overriding.
- Variables can be declared and initialized on the same line. Previously, variables were first declared (associated with a data type) and later initialized to a value. Additionally, the variant data type is no longer supported. However, the user still is not required to declare variables before initializing them. For example, the following ASP.NET Page directive can be used to turn off option explicit, thus suspending the need to declare variables before using them. Generally, this is a poor programming practice and is not encouraged.

  ```
  <%@Page Language = "vb" Explicit="false" %>
  ```
- Type Safety is used to generate errors when a data type conversion could fail at runtime.
- Error handling has been expanded to support the C–based language Try...Catch...Finally exception–handling statements. Visual Basic previously required, and still accepts, "On Error" statements in each procedure, which often results in duplicated code. The Try...Catch...Finally statements represent a more structured error–handling process that permits the nesting, and consequently the reuse, of exception–handling routines. Additionally, the Finally statement is a control structure for writing cleanup code that executes in both normal and exception conditions.
- Initializers can be used anywhere, including inside a control structure. Visual Basic .NET supports initializing the variable on the same line on which it is created. The semantics for a procedure–level declaration is similar if written in a single line as opposed to a declaration statement followed by an assignment statement. In other words, the following two examples are equivalent.

  ```
  Dim Y As Integer = 10

  Dim Y As Integer
  Y = 10
  ```
- Parentheses must be used in calling functions and subprocedures. All methods, functions, and subprocedures must now use parentheses to close the parameter list.
- Parameters are now passed to functions and subprocedures by value by default. Earlier versions of Visual Basic defaulted to passing parameters by reference. The ByRef keyword can still be used to pass a parameter by reference.
- New conversion methods are available for converting data types. Although older methods such as CStr are available, newer methods such as ToString are recommended.

17

These features combine with the .NET Framework to provide an easier environment to work with, a better performing application, and flexibility in incorporating Web–based applications. The resulting version of Visual Basic is more stable, scalable, and easier to use.

However, as previously noted, the downside to these much–anticipated new features is a lack of direct backward compatibility. Moderately complex applications cannot be ported easily from earlier versions to Visual Basic .NET. Although conversion tools exist, the number of required changes is significant. In many cases the migration tools simply flag an area that needs to be addressed, giving no indication of what the problem is or what needs to be done about it.

Visual C++

Among the programming languages Microsoft has altered to conform to the .NET Framework, Visual C++ has the fewest changes and requires the shortest learning curve. In addition, applications written in C++ can be ported easily to the new environment.

The most significant change to C++ revolves around the concept of "managed code," or the ability to compile code to MSIL for execution by the common language runtime. Within Visual Studio .NET, the developer can identify blocks of code as managed. The default option for Visual C++ development is still "unmanaged code," which is compiled directly to machine language code, lacks the type safety required for language interoperability, and forgoes the memory management features of the .NET Framework. In other words, Visual C++ developers can pretty much conduct business as usual. The language expands with a few additional items, which are detailed in the following list:

- Exception–handling tables
- Metadata describing the modules into which the code is compiled
- Location of object references (no need for CoCreateInstance; just use New to declare an object)

Visual C++ then uses managed code and lets the common language runtime handle the following items:

- Exception handling
- Security
- Lifetime management
- Debugging
- Profiling
- Memory management

C#

C# is Microsoft's programming language for its new .NET development environment. Microsoft's goal with C# is to provide a simple, modern, object–oriented .NET programming language that is Internet–centric. Although .NET code can be written in many languages, C# is the only language designed specifically for the .NET platform and for that reason may become the language of choice for this environment.

C# may be deceptively simple. Although it has only about 80 keywords and a dozen intrinsic data types, it is highly expressive. It includes support for all types of modern component–based, object–oriented development. C#, like C++ and Java, owes its origins to the C programming language. For that reason, C++ and Java developers will notice a striking similarity to those

languages and enjoy an easy–to–learn and familiar programming environment.

Specifically, C# is an elegant language object–oriented language that:

- Readily supports the definition of and working with classes and their properties and methods.
- Implements encapsulation, inheritance, and polymorphism, the three pillars of object–oriented programming.
- Features self–contained class declarations (definitions). Header files or IDF (interface definition files) are not required to define or publicize a class's signature.
- Supports the implementation of interfaces, or contracts, to implement functionality. While a class can only inherit, or extend, from a single class, it can implement multiple interfaces.
- Supports structs, which are lightweight custom types that require fewer resources than does a more complex, robust type defined as a class.
- Provides for operator overloading and modern exception handling.

Microsoft again has "bet the company" on a new technology and developed C# as its premier language for that technology. C# was developed for the .NET platform and was modeled after the success of earlier languages. It has a rapid application development environment courtesy of Visual Basic, performance due to C, object–oriented nature from C++, and elegance from Java.

ASP.NET

ASP.NET is a programming framework developed by Microsoft for building powerful Web–based applications. While not a programming language, ASP.NET is the cornerstone of the .NET platform's Internet–centric orientation. It is the latest version of Microsoft's ASP, a powerful, server–based technology for creating dynamic Web pages. Being a core element in the .NET platform, all .NET languages utilize ASP.NET as their entry point for Internet development.

Perhaps the biggest difference between earlier versions of ASP and ASP.NET is the way in which code runs on the Web server. With ASP.NET, code is compiled into executable classes that are compiled to native code by the common language runtime layer of the .NET Framework. All the earlier versions of ASP supported only scripting languages, which were interpreted. Since ASP.NET pages are now executable classes, they have access to all the other powerful features of the common language runtime, such as memory management, debugging, security, and strong data typing.

ASP.NET has built–in support for three languages: Visual Basic .NET, C#, and JScript.NET. The user can install support for other .NET–compatible languages as well. JScript.NET is Microsoft's latest version and implementation of JavaScript. Although this version still maintains its "scripting" feel, it is fully object–oriented and compiles to MSIL.

Because all ASP.NET code must be written in a .NET language, the ASP.NET programming framework is the foundation of Web services and Web Forms, the two components of ASP.NET Application Services. The following list illustrates how developers can exploit ASP.NET in developing Internet–centric applications.

- XML Web services gives developers the ability to access server functionality remotely. Using Web services, organizations can share their data and component libraries with other applications. Web services enable client/server and server/server communication through the HTTP protocol by using XML. Consequently, programs written in any language, using a component model and running on any operating system, can access Web services.

- ASP.NET Web Forms gives the developer the ability to create Web pages on the .NET platform. Web Forms enables developers to program embedded controls on either the client or the server. Using ASP Server Controls, Web applications enable the easy maintenance of state as information is communicated between the browser and the Web server.
- ASP.NET and the .NET Framework provide default authorization and authentication schemes for Web applications. Developers can easily remove, add to, or replace these schemes, depending on the needs of the application.
- Simple ASP pages can be migrated easily to ASP.NET applications. ASP.NET offers complete syntax and processing compatibility with ASP applications. Developers simply need to change the file extensions from .asp to. aspx to migrate their files to the ASP.NET page framework.

What .NET Means to Developers

Since the .NET Framework is a software development platform, it should be no surprise that .NET will have the biggest impact on developers. Here are some of the more significant benefits and implications awaiting them:

- With all the languages that soon will be compatible with .NET, developers have an unprecedented choice of interoperable languages to select from. That selection is far more than simply a lifestyle choice. .NET language choices provide the opportunity for a gradual transition from less complex languages to more powerful ones. Additionally, much of what is learned about the .NET base classes in one language carries over to the next language. Not only is the choice of languages unprecedented, so is the ease of the learning curve. Previously, Windows application programming required the developer to interact with components, application programming interfaces, and language–specific runtime libraries. This resulted in application development that was vastly different from one language to another. The new methodology, as described in Figure 1.3, allows all languages to interface directly with the .NET Framework.
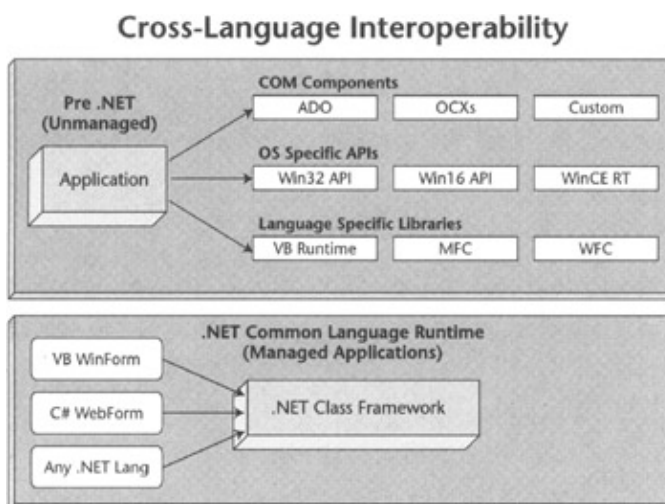
Figure 1.3: The .NET Framework overcomes previous weaknesses by allowing all languages to interface with the same common language runtime.

- The .NET languages are interoperable, meaning that classes developed in one .NET language can be freely reused and extended by a developer using another .NET language. While organizations may strive to standardize on a consistent development platform and language, developers may have more freedom to strike a proper balance between the skill required and the power necessary to accomplish their tasks.

20

- .NET continues the evolution of Windows development in the direction of providing more built–in functionality. The .NET base classes in particular encapsulate many of the mundane chores necessary in building application and network infrastructures. This frees the developer to focus more directly on solving business challenges.
- Developers may finally be freed from .DLL hell. Prior development environments stored information about components separately from those components. Consequently, installing and upgrading components were often a problem. .NET maintains these metadata inside the component itself, preventing the data from becoming out of sync with the component. In fact, different versions of the same component can coexist without compromising the integrity of the applications that are using them. Metadata about the component are maintained in the manifest and metadata elements of the components assembly, as shown in Figure 1.4.
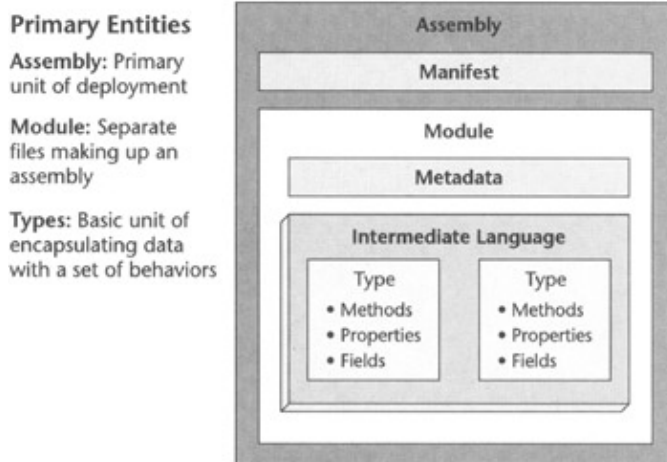
Figure 1.4: The anatomy of a .NET application includes three primary entities: the assembly, the module, and the types.

- The common language runtime now features a garbage collector, a process responsible for removing objects from memory that no longer have references to them. The developer is freed from the tedious and often fruitless activity of trying to assure that all objects have been destroyed.
- Finally, .NET developers will have the challenge of working in a true object–oriented programming environment. While object–oriented programming and design can be a complex process, the elegant implementation offered to developers in the .NET environment will make it a thrilling and stimulating challenge.

Review Questions

1. What are the three layers of the .NET Framework?
2. What is the purpose of the common language runtime?
3. What is the role of the base class libraries?
4. How do Windows Forms differ from Web Forms?
5. What are the significant changes to ASP?
6. Why should I use the .NET languages?

Chapter 2: .NET Business Solutions

Overview

The .NET Enterprise Servers combine to make a unique suite of products. Microsoft has positioned itself with a full line of products to serve the majority of application requirements. Application development traditionally has focused on the art of creating an application to perform a task or process. Although this is effective in getting the work accomplished, it often leads to decentralized, fragmented, and slow data storage solutions. Several departments may have their own applications for the same business process. Each department then maintains its own data and servers.

In most cases the organization realizes the need for analysis of the data. At this point the data are distributed throughout the organization. Sometimes the data are stored in different database architectures. This makes it particularly difficult to report on the actions of the entire organization. The process is further complicated when there is a need to interact with other organizations (business partners, vendors, and so on). You read about paperless environments. Several books have been written on streamlining processes and reducing data entry. While these strategies are good, the overhead for creating the paperless integrated process can be enormous.

The .NET Enterprise Servers are Microsoft's solution to this difficult task. The products integrate with each other around some common services. Several of these services (XML, COM+, and ASP.NET) are referenced throughout this book. Through common services it is possible to create solutions that take advantage of the features of multiple products while avoiding the overhead of incompatibility issues.

This chapter provides an introduction to the .NET business solutions. First the chapter outlines the goals of the .NET Enterprise Servers: What do they do well? How do they help you develop enterprise applications? The chapter then introduces the concept of enterprise application integration (EAI), the process of integrating the organization's applications to decrease business process duplication or centralize the reporting of the decentralized processes. The chapter then describes business–to–business (B2B) solutions. Many organizations would like to streamline their business processes with vendors and trading partners. The chapter then outlines business–to–consumer (B2C) solutions. Finally, the chapter summarizes the contents of each of the sections.

The goal of this chapter is not to provide the details of any given .NET Enterprise Server. Each server has a chapter dedicated to it later in the book. The purpose of this chapter is to lay a foundation for the types of business solutions you can create. The chapters that describe a server relate back to this chapter as a reference for the type of business solution that server facilitates.

.NET Enterprise Server Purpose

Microsoft has put together a suite of products that are intended to make enterprise development less complicated and more robust than Microsoft's traditional roots. The .NET Enterprise Server suite was created with the following goals:

- The application should be low–maintenance.
- The application should be secure.
- The application should scale well. As the number of users increases, the performance should not decrease significantly.

- The application should be stable.
- The application should integrate well with other applications and processes.

Each of these goals is described in more depth in the following sections, which reference products and features from an overview level. The reader should refer to the chapter associated with each product to find the details of implementation.

Low Maintenance

Nobody wants to waste time maintaining applications. The .NET Enterprise Server suite makes application maintenance easier than it was in the past. Many of the servers are deployed on Web servers. Applications that are Web–based are deployed and maintained at a single point, the Web server. If the users just need a browser, you do not have to be as concerned about their computers. In addition to the Web, Microsoft has released Application Center and Content Management Server. Many organizations already have deployed their applications to the Web.

The Active Server Pages (ASP) that make up the application typically connect to data that are stored in a database. As the number of users increased, the performance of the application decreased. Many companies realized that they needed more than one server. Microsoft implemented a load–balancing service that allowed organizations to deploy their Web content to more than one server and have the Web servers share the connection load. This was effective, but there was no built–in process to guarantee that the data were the same on all servers. Application Center 2000 was provided to synchronize the data across all Web servers. This ensures that all Web users see the same content. You only have to deploy the content to a single server, and the rest of the Web servers are automatically updated.

Security

As you port your applications to an enterprise level, there are several additional variables. You have more users that need to interact with your application, you have the Internet as a possible medium for application delivery, and you have multiple products that work together to provide a business solution.

The .NET Enterprise Servers make security a priority. You can implement security in each of the servers. Your application should account for security at the following layers:

- **Authentication.** The user should log on to the system, and the logon should be authenticated in a secure fashion. You want to avoid the passing of clear text usernames and passwords.
- **Authorization.** When a user makes a request of an application, the request should be validated. You should be checking to ensure that the user has the proper level of permission to perform the task at hand.
- **Encryption.** If the application is going over the Internet, you may want to encrypt the contents of the data that are being passed from the Web server to the browser and vice versa.
- **Protection.** If the application is ported over the Internet, you will want to give Internet users access to only a limited part of the network. You do not want Internet users to have access to all of your servers.

The .NET Enterprise Servers can be used to satisfy these application requirements. For example, you can use Commerce Server to help with the user authentication and data encryption. With Commerce Server, you can validate user credentials by employing a variety of methods. You will

want to choose the method that best fits your organization. For more information on the methods of authentication with Commerce Server, refer to Chapter 10, "Commerce Server." Commerce Server also can be used to require encryption through Secure Sockets Layer (SSL). This method for security forces users to access the data over an HTTPS connection, where the data are encrypted.

In addition to Commerce Server, for security you should get familiar with Internet Security and Acceleration (ISA) Server. ISA is Microsoft's firewall solution. You can use ISA to isolate portions of your application for Internet users without having to give the Internet users full access to the internal network. Traditionally, firewalls have performed packet filtering, which drops the packets that should not be allowed in your network. ISA Server can perform this function, but it also extends this functionality to include a feature named Server Publishing. With Server Publishing you can make a server available, behind the firewall, to Internet users through their interaction with the firewall. The Internet users never directly interact with the server that is available. They interact with ISA Server, which proxies the request back to the internal server. For more information on Server Publishing, refer to Chapter 14, "Internet Security and Acceleration Server."

The other .NET Enterprise Servers also have important security features. As you choose the products for your business solution, you should evaluate your security options to improve your overall application security.

Scalability

You application should continue to perform well as the number of users increases. Traditionally, client/server applications struggled in performance as the number of concurrent users increased. This occurred primarily because the service requests were being sent to a single server. A single server can do only so much.

The .NET Enterprise suite addresses scalability. For example, SQL Server 2000 and Exchange Server 2000, which are the primary databases in .NET Enterprise solutions, can now use much more memory then their predecessors could. The additional memory support allows these databases to accept more concurrent connections.

Additionally, the .NET Server platform includes a network load–balancing service that allows you to deploy your application across multiple servers that are grouped in a server cluster. The load–balancing service then alternates the requests among the servers. You can deploy multiple servers for handling the client connections. Application Center 2000 extends this service with component load balancing, which allows you to distribute components to multiple servers to handle DCOM requests.

Stability

The data storage of your application has to be a primary concern. You want to ensure that your application and data are always available. Stability and availability are tied together. Poor stability decreases the ability to make the application available at all times.

.NET Enterprise Servers also improve application stability. For example, Exchange Server and SQL Server fully support the cluster service of the .NET Server platform. Clustering allows you to provide a standby server option. If your database servers go down, users still have access to the data. Clustering provides stability at the data level. The load–balancing services provide stability at the connection level. The user connection comes to a Web or application server, which then makes the connection to the data on behalf of the user. Through load balancing and Application Center 2000 you can provide stability at the connection layer that is similar to the stability provided at the data

layer by the cluster service.

Integration

The largest advantage of the .NET Enterprise suite of servers compared with its predecessors consists of the new features and products that support integration. Most of the .NET Servers support XML, which provides a central language on which all applications can communicate. Through XML, your .NET applications can perform the following functions:

- Share data
- Transfer data over any medium, including the Internet
- Interact securely with your firewall
- Provide Web reporting and analysis tools

.NET Enterprise Servers allow your application to take advantage of all these features. You can integrate the .NET Enterprise Servers to create business solutions that previously were difficult to achieve.

For example, BizTalk Server is an integration server whose purpose is to provide a medium for applications to share data. This is particularly beneficial in dealing with vendors and trading partners. Through XML and BizTalk Server you can transfer data from your organization to another organization and transform the data where appropriate. More information about creating business solutions with BizTalk Server can be found in Chapter 12, "BizTalk Server."

Enterprise Application Integration

For organizations of any size, developing enterprise applications that integrate with their other systems can be daunting. As you develop enterprise applications, several considerations can easily become stumbling blocks, including some of the following:

- Several different software and hardware platforms
- Different versions of software
- Applications that have never been upgraded
- Customized applications that do not meet the company standard
- Vendor−created applications that are not consistent with the applications created by your organization

In the past, most large organizations ended up with a wide variety of software and hardware platforms. Each of those solutions was created in its own time to meet a business requirement. The difficulty comes when you try to integrate all these applications. In addition to the Microsoft languages you are using, you may have other applications that are written in environments such as the following:

- Java
- JavaScript
- COBOL
- C++
- Perl
- CGI

In addition to the language in which the program is written, an application may have a database management system to which it writes data. Some of the following databases could be used in your organization:

- DB2
- Oracle
- VSAM data files
- Sybase
- Informix

As you can imagine, the wide range of products makes it difficult to report and analyze data at an organizational level. The purpose of this book is to describe the .NET Enterprise Servers and their integration with each other. Many of the concepts will apply similarly to the other applications and databases you are using.

Most products are beginning to support XML. ODBC and OLE DB providers exist for many of these database management systems. Through these interfaces you can integrate these products with your Microsoft solutions. For example, you could use BizTalk Server to transfer and map data between your Oracle financial application and your SQL Server human resources package. BizTalk Server can use XML as the means of integration. SQL Server also has built−in integration technology. Through Data Transformation Services (DTS), you can use OLE DB providers to transfer data to and from virtually any database management system.

The applications that are the most difficult to interact with are those written by a vendor for which you do not have the source code. In many of these cases the data are stored in a schema that is not intuitive to a developer who did not create the application. In these cases the best bet for integration is to work through the vendor. You should work to find the tools that are available and then see the level of integration that is possible. Most applications can at least export data to a text file.

Integration of applications can be done at several different layers. You will have to choose which layer of integration is best for your scenario. As shown in Figure 2.1, application integration can be done at the presentation, business, or data source level. The following sections outline the differences among these approaches.
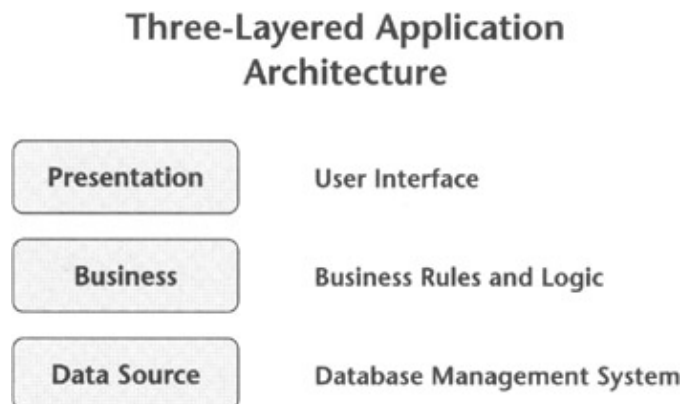
Figure 2.1: Each application has a logical three−layer architecture. Applications can be integrated at any of these layers.

Presentation Layer Integration

Presentation layer integration involves the combining of two application or business processes at the user interface level. Although this approach has its advantages, it requires a new design and requires that all the individuals involved communicate on the same page. Several organizations are choosing to implement this type of integration at the Web server level.

Let's look at an employee time tracking solution as an example. Currently, your organization has several methods of implementing employee time. In some areas of the company the employees punch time cards and hand them to a supervisor. The supervisor then enters the cards into a database or sends them off to human resources, which enters the time in a database. Other departments have a card–swiping system. When the employee comes in, the card is swiped and the data are recorded in a database. In yet another department the employees enter their own time via a Visual Basic application that communicates with the database. In the interest of consistency and for ease of use you would like to go to an intranet–based tracking system. You create the application by using ASP.NET, and the ASPs are installed on a Web server. Employees, as part of the logon process, enter their information at the Web site. You could have employees manually update the site at logon or logoff, or you could script the process and include it in logon and logoff scripts.

In this example, you have provided a single presentation layer for the business process. The new interface had to be created, but the process is consistent and uniform throughout the organization. Often the largest obstacle to presentation layer integration is politics. For this method to work, everyone has to be on the same page. The common solution for this layer of integration is via a Web server. In the Web server model, the presentation layer resides at the Web server and the application has to be maintained only on one machine.

Business Layer Integration

Integration at the business layer may require some intrusive measures in your applications. Your goal at this layer is to share business logic in multiple applications. For example, if you are in a bank, you have to manage transactions. A common transaction could be moving money between accounts. You might have an application that customer services reps use, one that customers use over the Web, and one that management uses to analyze data. Each of these applications has the ability to transfer money. Currently, each application maintains its own code to perform these transactions. You could centralize the transactions and let each application call the central function to perform the transaction. This provides consistency and less overhead to the developer of each application. In this model you are trying to reuse code as much as possible. This strategy also simplifies maintenance and later updates. When changes to the business logic are implemented, the code requires change only in one place rather than in several separate applications. The disadvantage of this model is that each application that currently needs the transaction has to be updated.

You can implement this level of integration through any of the following methods:

- **Using stored procedures at your database server.** The stored procedure can be executed from each application.
- **Creating a distributed object using DCOM or CORBA.** The object can be called from any application.
- **Implementing the logic at a transaction or application server.** A middle–tier server is added. The user interface interacts with the middle–tier server; the middle–tier server then makes the requests to the databases.

27

- **Sharing services as Web services.** Each application can make use of the Web services. More information on Web services can be found in Chapter 15, "Web Services with .NET."

This level of integration is common for applications that are currently in design. It is also effective as a moving−forward position. This position means that the choice of business layer integration is made and implemented in all applications from this time forward. Because of the overhead to existing applications it is not common for all applications to be updated. When the application is redesigned, the effort is made to implement business layer integration.

Data Source Layer Integration

Data source layer is the most common method of integration. You are taking advantage of your ability to open the data sources and share or move data. Each application still has its own presentation and business layer. Applications write their data to a database. At this point you can use an integration product to integrate the data. You can use OLE DB, ODBC, or XML to retrieve and store the data. Through data source integration you can do the following:

- Transfer data from one database server to another
- Update data based on the information in another database server
- Transfer data to a data warehouse for reporting and analysis

This model is the most common because it is the easiest to implement if applications are already in place. The other two options for integration are beneficial if you can redesign and redeploy an application. They are, however, much more intrusive to existing applications. You may find it easier to recreate an application than to integrate at the presentation or business layer. The data source layer is different. Because you are dealing with the data, you can transfer data back and forth without affecting the code that manages your presentation and business layers. The downside to data source integration is that it does not encourage standards and consistency. When you integrate at this level, you continue to support applications that have different architectures, and everyone is generally not on the same page.

You can perform data source integration in a number of ways. The .NET Enterprise Servers can help with this task. For example, SQL Server includes a tool named DTS that you can use to transfer and transform data from one source to another. DTS has full support and drivers for most database management systems. It also has many built−in providers for nonrelational data sources such as Excel (.csv) files and text (.txt) files. You could also use BizTalk Server to integrate data using XML. You could transfer and map data between any two data sources that are XML−compliant.

Summary

EAI solutions can be challenging and daunting. You need to choose the level of integration that is most appropriate to your scenario. If you have control over the development of the applications and time to implement your solution, presentation or business layer integration is preferred. In many cases you won't have the time or control, and so data source integration is the only option.

Business−to−Business Solutions

Business−to−business solutions are not as common as EAI solutions. Many organizations have not tried to implement these solutions. Several large organizations have tried to tackle this in what has

been referred to as the electronic data interchange (EDI) solution.

In B2B solutions, you are integrating your business process with another organization's business process. Traditionally, most of this occurs via paper. There are systems that are responsible for sending and tracking invoices. When you receive invoices and other correspondence from another organization, you typically have an individual enter them into your system, and then you start tracking them. In most cases, both companies enter the information manually and track the status of the correspondence individually.

Organizations that are highly dependent on each other may invest in a dedicated line between two organizations. In this solution the vendor or trading partner is given access to required applications on your network. While this solves the problem, it can be costly for both companies.

With .NET Enterprise Servers you can integrate your applications with your vendors and trading partners. The organizations you do business with the most may appreciate your attempt to streamline the business processes. This type of integration can follow a couple of different approaches.

The first and most common approach is integration through a Web site. Many organizations are creating business–based Web sites for vendors and trading partners to use. While this solution decreases your data entry, it does not fully integrate the two business processes.

For example, let's look at a basic invoicing system. You are in a large organization that receives hundreds of invoices daily from multiple vendors. You would like to decrease the overhead of data entry related to those invoices. You create a Web site that vendors and trading partners can log into and use to enter their invoice information. The Web site is created using ASP.NET, and you store the ASPs on Internet Information Server (IIS). After a vendor enters the invoice, the ASPs connect to your SQL Server–based invoice database to store the information. Now the data are in your system without the overhead of data entry. The downside to this model is on the vendor side. While the vendor may appreciate the solution because it reduces mail time, the vendor still has to enter the information manually in its own system and into your system through your Web site.

The second approach, as shown in Figure 2.2, provides a fully integrated solution. You can use BizTalk Server to transfer the invoice information from the vendor's database to your database. BizTalk Server has the ability to transfer the data via XML, which allows you to take advantage of the Internet as a transfer medium. With this solution, the data are entered once and both parties can use the information.
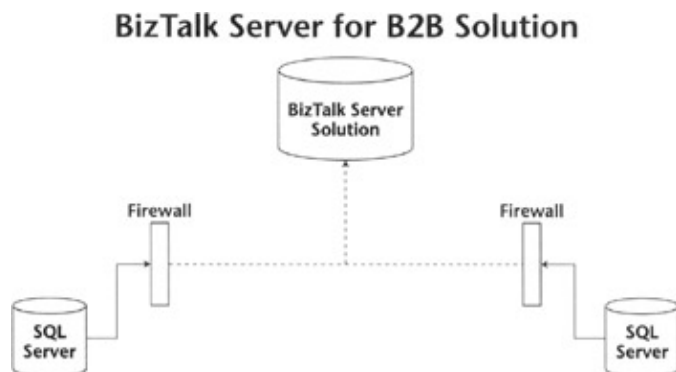
Figure 2.2: Businesses can integrate business processes by using BizTalk Server.

Business–to–Consumer Solutions

More and more organizations would like to use the Internet to boost profits. The Internet provides marketing opportunities that previously were limited and costly. You can make your products and services available to the entire world. With business–to–consumer solutions you can create a Web presence that promotes your products in a secure, fast, and reliable manner. The .NET Enterprise Server facilitates the use of the Internet as a medium for selling your solutions.

For example, as shown in Figure 2.3, your Web site can be developed in ASP.NET and deployed on IIS. The ASPs that are stored on IIS can then connect to an SQL Server database to get a list of products and store customer information. Typically, your Web server is outside the firewall and the SQL Server database is inside the firewall. You can further enhance your Web presence by using the following .NET Enterprise Servers:

- **Application Center 2000.** The load–balancing and synchronization services of Application Center 2000 allow you to deploy your site on multiple Web servers. The Web requests are then load–balanced between the servers. This product helps increase performance, stability, and maintainability.
- **Commerce Server 2002.** You can use Commerce Server to enhance the security at your site. You can guarantee a secure, encrypted connection from your Web users to your site.
- **Internet Security and Acceleration Server.** ISA Server is Microsoft's firewall. You can use ISA to keep your database protected from Web users.
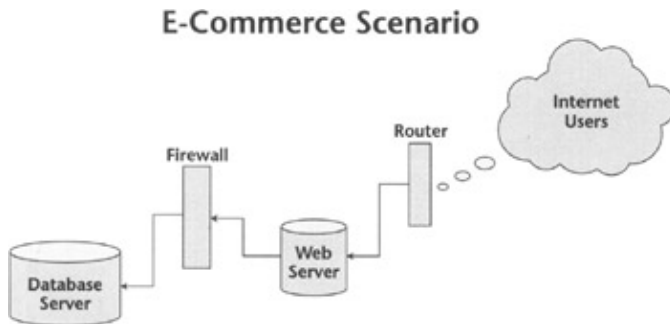
Figure 2.3: Common e–commerce Web site scenario.

Summary

As an enterprise developer, you have the daunting task of integrating current applications with your organizational directives for the future. Through this process you still need to make sure that your applications are fast, secure, stable, and—most important—easy to maintain. The .NET Enterprise Servers can help with this task.

You can use the servers discussed in this chapter to perform EAI, B2B, and B2C solutions for your organization. Later chapters outline the technologies and products that make this process possible.

Review Questions

1. What factors should you consider in creating an application?
2. What are the three layers on which you can perform EAI?
3. What is the advantage of data source integration?
4. What is the disadvantage in implementing integration at the data source level as opposed to the presentation or business layer?

5. What is the role of BizTalk Server in B2B applications?

Chapter 3: Role of .NET Enterprise Servers

Overview

A primary goal of the .NET initiative from Microsoft is to make data available at any time, at any place, and on any type of device. To support this goal, Microsoft introduced the .NET Framework and made significant changes in the underlying operating systems. The primary architectural change has been the move from the traditional client/server architecture to the Web services of .NET. This switch in methodology has changed the focus of application development from the local machine to a server–based presentation layer.

The change in this architecture has brought about several changes in what has been known as the BackOffice Servers. The new servers now known as the .NET Enterprise Servers have upgraded some of the old servers (SQL Server and Exchange Server) and include several new server products (BizTalk Server, Application Center Server, and Content Management Server, among others). All these servers are founded on the COM+ technology and have tight integration with XML.

This chapter provides a description of the role the .NET Enterprise Servers will play in the future. The initial section describes the features that are common to all the servers, and then the chapter moves to an introduction of each of the .NET Enterprise Servers. The introduction of each server describes, in an overview section, the design goals of the product, followed by the features of the product that meet the design goals. The chapter begins with an overview of the new .NET Enterprise Server operating system. It then moves to a description of Exchange Server, SQL Server, Application Center Server, Commerce Server, Content Management Server, SharePoint Portal Server, BizTalk Server, and finally Internet Security and Acceleration Server (ISA).

The purpose of this chapter is to provide a direction for the use of the .NET Enterprise Servers. Each server product is discussed in a chapter later in the book; this will provide deeper coverage of the use of the development features identified in this chapter. For more information on specific features referenced in this chapter, refer to the chapter corresponding to the product you are reading about.

Common Features

Microsoft created the .NET Enterprise Servers to leverage the interoperability of the servers. Developers now have all the tools necessary to design, create, deploy, and manage enterprisewide software applications. In describing the common features in the different .NET Enterprise Servers, this section first addresses the support for common protocols and standards and then moves to application scaling and finally to interoperability.

Common Protocols and Standards

Each of the .NET Enterprise Servers now can use XML Web services and its underlying data format. The data can be stored and retrieved in XML format. XML also can be used as the format for interoperability between the servers. Internally, each of the servers may translate the data to its core language, but they all have the ability to expose the data by using XML. The exposed data allow for interoperability between the .NET Servers and other products that are compatible with XML. Each of the servers also supports full interaction with HTTP, HTML, FTP, and XPATH. This allows the free exchange of data between the servers, over the Internet, and through the firewall infrastructure. While all the servers maintain their own identity and core languages, they all provide

common support for the Internet standard protocols.

In addition to supporting the common protocols mentioned previously, the .NET Enterprise Servers are all deployed across the same platform. All the .NET Enterprise Servers are deployed on either the .NET Server platform or the Windows 2000 platform. Because the servers are deployed on the same operating system, they all adhere to the set of requirements set by the operating system. This allows the servers to interface along the common operating system settings. Throughout this book the integration features of the .NET Enterprise Servers are made easier because these servers are deployed on a common operating system.

Application Scaling

One of the primary design goals of the .NET Servers is to provide scalability. Scalability means that an application may grow in scale without a decrease in performance. Microsoft has attempted to create an infrastructure that allows applications to scale as large as necessary. As the applications have grown, the primary focus has been on maintaining and in some cases increasing the performance of each application. In the past, as applications became larger, the applications generally slowed down. This resulted in decreased response time and increased overhead to the servers. Now, with the .NET Enterprise Servers, applications can scale with larger amounts of data spread across multiple servers without suffering performance degradation. Users may be accessing a Uniform Resource Locator (URL) that directs them to any of a hundred servers, and the users do not have to know that more than one server is involved. Applications can scale out across multiple servers and use the load–balancing features to guarantee the same level of performance.

Through the .NET Servers, developers can take advantage of server farms and clusters. The farms allow you to spread the application and ensure that the process is transparent to the user. Clusters allow you to guarantee the availability of the application. If a server goes down, the application is directed to another server that has picked up the load. More information on server farms and clustering can be found in Part Three of this book, "Enterprise Solutions with .NET Services."

In addition to scalability .NET Enterprise Servers provide deployment and management tools to make the process easy to implement. Although many of the options presented in this book are not new, they are much easier with the .NET Enterprise Servers suite of products. You can now manage multiple servers from a single console and manage all the tiers of an application from the same interface.

Interoperability

The applications that are developed may need to use multiple products. The developer must achieve the goal of creating a system that brings the users together with the logic and data of the application. In addition to providing the system to match the business process, often the system has to be monitored and the results reported. These requirements may be best met by coordinating different products and applications so that they work as one. Through the support of common standards and Internet protocols, the .NET Enterprise Servers can be integrated seamlessly to provide a deployable solution.

The .NET Enterprise Servers rely on the operating system to perform many of the application core features. Before the .NET Enterprise Suite, developers had to build caching, clustering, and component load–balancing features into the application. This required significant overhead for the developer and the application. Now these features have been pushed to the operating system level to allow the developer to devote more time to the business logic of the application. This also allows multiple server products to use the same features and interact with the same processes. This helps

ensure consistency by requiring each of the .NET Servers to act in the same manner when using certain operating system features, such as clustering and load balancing.

.NET Server

Built on the reliable Windows 2000 server family, Windows .NET Server integrates a powerful application environment to develop innovative XML Web services and business solutions that dramatically improve process efficiency.

.NET Server is the operating system on which all of deployable solutions presented by this book are installed. .NET Server is Microsoft's operating system upgrade to Windows 2000. .NET Server was designed to scale much larger than its predecessor. With this new operating system, Microsoft has attempted to focus on improving scalability, stability, and performance. .NET Server ships in four platforms, as detailed here. For more information on each of these servers the reader should refer to Microsoft's .NET Server home page at www.microsoft.com/windows.NETserver.

- Windows .NET Web Server is designed for Web serving and hosting, providing a platform for rapidly developing and deploying Web services and applications. .NET Web Server fully supports the .NET Framework and most Internet services, such as Internet Information Server (IIS), ASP.NET, and IPv6 (the next generation of TCP/IP). The .NET Web Server is limited in scalability because it only supports up to 2 GB of memory and two processors. It also does not include several features, such as Terminal Services, Clustering Services, Remote Installation Services, MetaDirectory Services, and support for 64–bit processors.
- Windows .NET Standard Server is designed for the everyday needs of businesses of all sizes, providing a solution for file and printer sharing, secure Internet connectivity, centralized desktop application deployment, and rich collaboration between employees, partners, and customers. The Standard Server fully supports Terminal Services, Active Directory, and Windows Media Services. .NET Standard Server supports up to 4 GB of memory and up to two processors. It is limited in that is does not support Clustering Services and MetaDirectory Services and does not provide support for 64–bit processors.
- Windows .NET Enterprise Server is designed for the general–purpose needs of businesses of all sizes. Enterprise Server is the platform of choice for applications, Web services, and infrastructure, delivering high reliability, performance, and superior business value. .NET Enterprise Server scales up to 64 GB of memory and up to eight processors. It includes all the features of the Standard Edition and also supports Clustering Services, MetaDirectory Services, and the 64–bit Itanium processor.
- Windows .NET Datacenter Server is designed for business–critical and mission–critical applications that demand the highest levels of scalability and availability. .NET Datacenter Server scales up to 128 GB of memory and up to 32 processors. It also supports four–node clustering, whereas .NET Enterprise Server supports just two–node clustering. For more information on the Clustering server of the .NET Server platform, refer to Chapter 18, "Clustering .NET Servers."

.NET Server provides a solid platform to deploy Enterprise solutions throughout your network. The product has been built to provide developers with new opportunities to build large–scale applications seamlessly. Figure 3.1 identifies the .NET Enterprise Servers that can be deployed on the .NET Server platform.

.NET Enterprise Servers deployed across the .NET Server platform
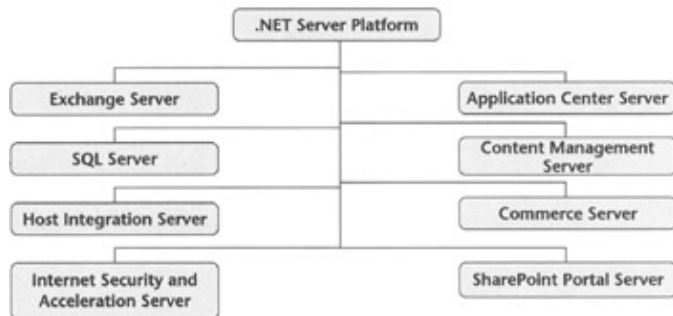
Figure 3.1: NET Enterprise Servers are deployed on the .NET Server platform. Many of them can be deployed on earlier versions of Microsoft operating systems as well.

The next several sections describe the .NET Enterprise Servers that are deployed on the .NET Server platform. Each of these server products also has a dedicated chapter later in the book to further describe its Enterprise development features.

Note The .NET Server platform is not the primary topic of this book. If you need more information or technical support on the configuration of the operating system, go to the Microsoft Web site at http://www.microsoft.com/. This book addresses and presents the solutions that can be created on this new operating system. The next several sections introduce the .NET Enterprise Servers, which can be deployed across the .NET Server platform.

Exchange Server

As a member of the .NET Enterprise Server family, Microsoft Exchange provides an email infrastructure combined with a powerful collaboration environment. Exchange is not a new product in the .NET Enterprise Server family. For years Microsoft Exchange has provided the email infrastructure for many organizations. Exchange Server 2000 provides many enhancements compared with its predecessor. Exchange Server 2000 gives information technology (IT) system administrators a product that defines the industry standard for email scalability and stability while providing developers with a platform to easily store and retrieve data useful for collaboration and workflow applications.

Design Goals

Microsoft Exchange 2000 meets the following design goals:

- Provides an email product that integrates seamlessly with Active Directory and the other features of the Windows Server platform.
- Provides a collaboration platform that can easily share and retrieve data used in workflow applications.
- Increases the flexibility of application development by allowing developers to interact with data stored in Exchange through a variety of data access methods.

Features

Microsoft Exchange 2000 provides the following features to both IT administrators and developers to address the design goals listed in the preceding section:

- **Integration with Active Directory.** Exchange Server no longer has its own directory. All directory information is stored and shared with Active Directory. The exchange object properties are stored and configured with the Active Directory objects.
- **Integration with Internet Information Server.** The majority of mail transfer in Exchange 2000 is performed via Simple Mail Transport Protocol (SMTP). Exchange no longer has its own SMTP service. Exchange uses the SMTP service of IIS to transfer its messages. Through the SMTP service of IIS, developers can create event sinks, which attach an ActiveX script to a transfer event in SMTP. Event sinks allow for the development of custom email filters to help prevent viruses.
- **Web Storage System.** The Web Storage System consolidates the file system, database, and collaboration services into a single server system. You can access the Web Storage System through nearly any application. The Web Storage System relies on the .NET Server and IIS to support Web browsers and WebDAV applications by means of HTTP, WebDAV, and XML. You also have the ability to interact with the Exchange Server database by using standard database languages such as ADO.NET and Transact–SQL. You do not need to perform any additional configuration to provide access to user mailboxes and public folders through any of the interfaces mentioned here.
- **Collaborative Development.** Collaborative Data Objects (CDO) for Exchange Server 2000 is a technology for adding messaging solutions to the application infrastructure. You can manage messages, folders, contacts, and other items through CDO. CDO combines with ADO.NET to unleash the potential for developers to interact with the data stored in Exchange Server in ways that previously were not available.

SQL Server

Microsoft SQL Server 2000 is also a member of the .NET Enterprise Server family of products. Among all the products described in this book, SQL Server has the fewest changes from its predecessor. SQL Server is Microsoft's relational database management system (RDBMS) and is used to create, deploy, and administer databases. This is Microsoft's scalable database solution, which can be used as the back–end database for the deployment of scalable, mission–critical client/server, Internet–based, and *n*–tier applications.

Design Goals

In developing SQL Server 2000, Microsoft realized that it already had many of the core features necessary for a quality RDBMS in SQL Server 7.0. It decided to provide enhancements to the product that would make the product more scalable and stable for larger–scale applications without taking away any of the features that made the product competitive. Microsoft wanted to accomplish the following design goals when creating this product:

- Increase flexibility in data access. Specifically, it provides an easy interface for Web access.
- Increase the scalability and functionality of the Online Analytical Processing (OLAP) services.
- Provide more scalable database solutions.
- Provide more stable standby server options.

Features

SQL Server 2000 provides the following features for IT developers and administrators. These features can help provide a more stable, scalable, mission–critical application development

environment. Although SQL Server includes many other features, the following features facilitate Enterprise development solutions with the product.

- **XML access to SQL Server.** SQL Server 2000 allows you to store and retrieve data by using XML. XML provides a standard formatting and data representation language. XML is a key component in the future of application–to–application data transfer. It also can be used to publish SQL Server data to the Internet.
- **Analysis Services.** SQL Server 2000 includes Analysis Services, which is the upgrade to the OLAP services of SQL Server 7.0. Analysis Services provides the core of the data warehousing framework and has been expanded to support data mining, XML, and better interaction with data transformation services (DTS). Data mining is used to find patterns and relationships between data that otherwise might not be noticed. DTS are the services provided with SQL Server 2000 to transfer data among multiple data sources. In Analysis Services, Microsoft has created an industry–standard data–warehousing tool.
- **Federated Database Servers.** SQL Server overcomes many of its scalability concerns by providing the option to create Federated Database Servers. This functionality allows you to partition a single database across multiple servers. The partitioning is transparent to the user but allows the application to grow to as large a size as needed. To the user the database appears as though it were stored in a single location whereas it actually is spread across multiple database servers.
- **Integration with Microsoft Clustering Services.** SQL Server 2000 is cluster–aware, meaning that it understands the cluster architecture provided by the .NET Advanced Server or the .NET Datacenter Server. By using clustering services, you can create a true fault–tolerant solution. Multiple servers (up to four) can be processing, and when a server fails, a failover occurs (failover is the process of moving the load of one server to another service). The failover is quick (approximately a minute) and seamless to the user. The other server takes over the processing, and no changes have to be made to the front–end applications.
- **Log shipping.** Log shipping allows you to constantly dump and copy transaction log backups from a source server to a destination server and then load those logs onto the destination server. It provides a warm backup server solution. You can use the destination server in a read–only mode to help facilitate large queries or analysis work.
- **Multiple instances of SQL Server.** With SQL Server 2000 multiple instances of SQL Server can be installed on a single machine. This feature allows each instance to run its own services and security configuration. You can separate the databases for different applications into separate instances so that the stability of one application does not affect the stability of another application. You also can identify security on an instance–by–instance basis, which allows for an application to have a database administrator who does not need to have full permissions to use other databases that reside on the same server.

Application Center 2000

Application Center 2000 is a tool for creating, deploying, and managing Web–and component–based applications. Typically, these are line–of–business applications that require a high level of availability and need to provide acceptable response time. The servers hosting these applications are expected to handle traffic that is characterized by high volumes, exponential growth curves, and load fluctuations.

Good performance, of course, is desirable in any product. In addition to offering optimal load–balancing algorithms for different types of applications, Application Center Server provides

tools for monitoring system performance and allows the system administrator to adjust load on a server–by–server basis. This approach recognizes the realities of heterogeneous server farms and provides far greater flexibility than does a one–size–fits–all approach.

This is a new product in the .NET Enterprise Server family and should be useful in providing the following cost benefits:

- *Manageability* through a centralized management console that is minimal and familiar. This console is used to organize and manage replication, load balancing, and the monitoring of Web and COM+ applications.
- *Scalability* that is both linear and flexible. Additional servers can be added to a cluster as needed to accommodate seasonal peaks and can be removed (and reallocated within the organization) as the load decreases.
- *Reliability* by eliminating the single point of failure associated with scaling up or hardware–based load balancing. Application Center Server also transparently removes a server from operation if there is a hardware or software failure.

Design Goals

In addition to providing the benefits itemized in the preceding section, Application Center meets specific design goals:

- Provides easy administration of Web and COM+ applications on multiple server groups.
- Provides a user interface that is easy to use, browsable, scalable, searchable, and minimal.
- Increases the discoverability of the most frequently used settings; buries or eliminates advanced settings.
- Provides easy access to related partner configuration tools (i.e., Microsoft IIS, COM+, and Microsoft Health Monitor).

Features

The following features of Application Center Server achieve the design goals and application requirements described previously. Figure 3.2 identifies the core features of the Application Center Server. The following items describe the role of each of the features described in that diagram.
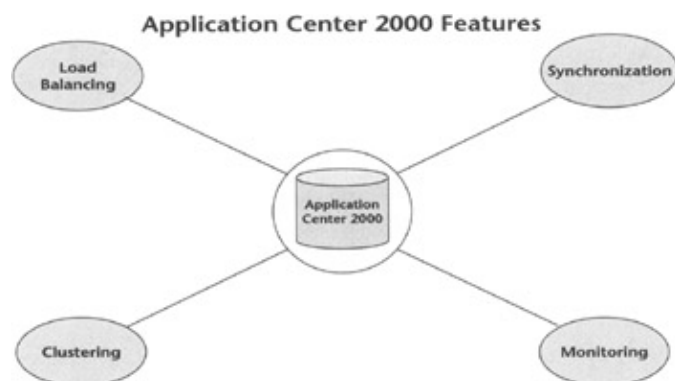
Figure 3.2: Application Center is the core for several features that are attractive to many Web developers.

- **Load balancing.** Application Center supports both network load balancing, which is integrated with the .NET Server platform, and component load balancing. This allows the developer to scale an application across multiple servers. As the number of servers and

components grows, your application performance does not have to suffer.

- **Cluster services.** Application Center Cluster services are used to group servers. You can then customize settings, content, and applications across the cluster. Content can be deployed either within a cluster or to another cluster, providing unlimited flexibility in deploying Web content and applications.
- **Monitoring.** Application Center supports real–time event, performance, and health monitoring. This is useful in tracking the performance, activity, and uptime of Web sites.
- **Synchronization.** Application Center 2000 can synchronize the content of a Web site across multiple servers. For load–balancing scenarios, this helps guarantee that all servers in the cluster have identical content.

Application Center Server also offers high availability. Requests and transactions can be rerouted automatically to another server if one of the servers fails. This builds in fault–tolerance features and allows applications to continue working after most common failures.

Commerce Server

Microsoft Commerce Server is a comprehensive, integrated commerce solution that provides ready–to–use features and tools to quickly develop, deploy, and manage commerce applications for the Web. You can use Commerce Server to build scalable business–to–consumer (B2C) and business–to–business (B2B) site applications.

Commerce Server provides business managers with the ability to easily analyze and manage business operations. It includes sophisticated analytical capabilities and a real–time feedback loop that empowers business managers to respond to the changing needs of customers and partners. By creating highly dynamic, personalized commerce sites, you can optimize the customer experience, encourage repeat business, and compete successfully in the e–commerce marketplace.

Design Goals

In creating Commerce Server, Microsoft had to analyze the needs of flexibility for the developer and integration with other products while trying to provide a fast and easy way to build a commerce solution. This product was created with the following design goals in mind:

- Provides a secure, fast, and stable commerce solution for Web developers.
- Provides a product organization system to help maintain the current products your company is offering through the commerce solution.
- Provides a solution that integrates with the other .NET Enterprise Servers.
- Provides a commerce solution that is easy to use, implement, and maintain.
- Increases the scalability of current e–commerce solutions.

Features

Commerce Server features function together seamlessly, enabling you to provide merchandising, catalogue display, customer service, and orders capture and receipt. Table 3.1 identifies the features that are provided with Commerce Server along with a description of what each feature can do for you.

Table 3.1: Commerce Server Features

COMMERCE SERVER FEATURE	DESCRIPTION
Product Catalog System	Create and update catalogs of products to sell.
	Import and export catalogs to and from other data sources.
	Browse for products by category, property, or free−text search.
	Apply discounts to products.
Profiling System	Manage millions of users and organizations.
	Enable users to manage their own profile information and research the status of their orders.
	Build profiles for any business−related entities, such as users and organizations.
	Determine what information to collect about users visiting your site.
	Analyze profile information to determine who is visiting your site.
	Create advertising, discount, and direct mail campaigns to target specific user groups.
Business Analytics System	Import large amounts of site usage data collected from different data sources into the Commerce Server data warehouse and manage the data.
	Analyze site effectiveness by running Commerce Server reports or creating your own custom reports.
	Identify specific groups of users by running a report and then exporting the results to a list. Use the list in a direct mail campaign or to update the properties of the users so that you can target merchandise or content to them.
	Review collected data to identify hidden trends and new customer segments and to view the properties of the segments.
Targeting System	Personalize the buying experience with targeted merchandising.
	Deliver the optimal content for a given user in a given context.
	Create, analyze, and manage personalized and targeted discounts and direct marketing and advertising campaigns.
	Add predictive capabilities to your Web site.
	Target ads or discounts to users of a specific profile. For example, you can target ads to female users who visit your site three times a week and have purchased three or more products.
	Create and schedule campaigns for customers who compete within the same industry. The competing ads are never shown on the same page.
	Charge your advertising customers on the basis of the number of ad requests or clicks they want their ads to receive.
	Charge your advertising customers on the basis of the page on which they want their ads to display.

Content Management Server

Microsoft Content Management Server 2001 and Microsoft Commerce Server combine to build personalized and content−rich B2B and B2C sites. Microsoft Content Management Server provides content contribution and delivery and site management capabilities, and Commerce Server includes an application framework, sophisticated feedback mechanisms, and analytical capabilities.

The seamless integration of Content Management Server automated content publishing, workflow, and content–sharing and reuse capabilities, combined with the Commerce Server product catalog, profiling engine, transactional processing, and shopping cart, allows you to deploy powerful e–commerce sites quickly. By integrating the features of Content Management Server 2001 with those of Commerce Server, organizations realize the benefits of:

- Automated publishing and workflow for Web content
- Centralized control of site design and usability for rich product catalogs
- Quick and cost–effective deployment of content–driven sites
- Delivery of targeted and personalized experiences to site users
- Effective use of technical and business skills

Design Goals

To fulfill the needs of companies and their customers, e–commerce sites must be effective, empowering, dynamic, and fast. Customers demand personalized information to make decisions. Companies require cost–effective and agile solutions.

Most Internet users go online to find information to get a job done or to help them make purchase decisions. They want information personalized to their needs and interests. E–commerce sites that do not recognize these user requirements will fail. To be truly effective, e–commerce sites must:

- Attract potential customers
- Convert browsers into buyers
- Retain existing customers

Two key methods for achieving these goals are providing content with product catalogs and personalizing content for groups and individual users.

In addition to empowering consumers by providing robust information and a personalized browsing experience, sites must empower business users. Business users need to create profiles and target content for specific users and customer groups. Business users also must be able to publish their own content—including rich product–catalog pages, news, and articles—using familiar tools.

To maintain a high level of personalization, e–commerce sites must be dynamic. Personalized sites cannot be well served from a flat–file system. Instead, sites must be assembled from a dynamic database as users request pages. Without this dynamic functionality, the cost of maintaining a highly personalized, content–rich site is prohibitive.

Sites must be fast to deploy and fast to change. Sites must be built with tools familiar to your team and must use their skill sets properly. Business managers must be in control of site content and customer profiling. They must be able to make changes to content, special offers, and user profiles based on user feedback and site analysis. Web developers and designers must be in control of site usability, architecture, and design.

Features

Using Content Management Server together with Commerce Server enables organizations to meet the needs of the essential e–business challenges discussed in the preceding section. This integrated solution enables organizations to:

- Create and manage rich product catalogs

- Deliver audience–targeted and personalized content
- Quickly build manageable e–commerce sites
- Effectively use team skill sets

The following features are supplied with Content Management Server to meet these design goals:

- **Content contribution for nontechnical business users.** Browser–based content creation and contribution features a comprehensive and easy–to–use tool set for business users to manage their own content.
- **Role–based workflow and approval.** Built–in workflow and approval process ensures that content is always accurate.
- **Personalized and targeted content.** Dynamically rendering pages as requested provides real–time personalized content. Easily tagged content with metadata for classification and personalization assists in this process.
- **Content reuse across applications.** Content is stored and managed separately from the design elements of the site, allowing for diverse ranges of content reuse. For example, content can be served to different devices or integrated with other e–business initiatives.
- **Rapid time to market.** Industry–standard technology and a complete publishing Application Programming Interface (API) enable rapid application development.
- **Enterprise scalability and reliability.** Solution scales vertically and horizontally for enterprisewide Internet, intranet, and extranet applications.

SharePoint Portal Server

SharePoint Portal Server is a flexible solution that integrates search and document management with the tools you use every day. SharePoint Portal Server works with Internet Explorer, Microsoft Office applications, and Web browsers to help you create, manage, and share content.

Design Goals

SharePoint Portal Server was created to expand the functionality of the tools end users work with on a day–to–day basis. Microsoft provides the following advantages to users through this product:

- Easy search ability regardless of the location of the data
- Easy security management for Web documents
- Versioning of Web documents to keep track of multiple saved versions of documents

Features

SharePoint Portal Server offers the following features to assist in the deployment and use of user–friendly Web sites:

- **Publishing on a dashboard site.** A dashboard site provides access to information stored inside and outside an organization. Through a dashboard site, users can search for and share documents regardless of location or format.
- **Searching across multiple locations.** SharePoint Portal Server creates an index of searchable information that includes all workspace content. It also can include a variety of information stored outside the workspace on other SharePoint Portal Server workspaces, Web sites, file systems, Microsoft Exchange Servers, and Lotus Notes databases.
- **Document access based on user roles.** By using role–based security, SharePoint Portal

Server helps you control access to documents in the workspace.

- **Version tracking of multiple documents.** Check–in and check–out ensure that only one person works on a document at a time. Versioning keeps archived versions for reference and recovery.
- **Document routing for review and approval.** This feature determines when documents are visible to readers on a dashboard site.

BizTalk Server 2000

A member of the Microsoft .NET Enterprise Server family of products, Microsoft BizTalk Server 2000 unites in a single product enterprise application integration (EAI) and B2B integration. BizTalk Server 2000 enables developers, IT professionals, and business analysts to easily build dynamic business processes that span applications, platforms, and businesses over the Internet.

Microsoft BizTalk Server 2000 provides a powerful Web–based development and execution environment that integrates loosely coupled, long–running business processes both within and between businesses. BizTalk Server can handle transactions that run as long as weeks or months, not only minutes or hours.

The server provides a standard gateway for sending and receiving documents across the Internet as well as providing a range of services that ensure data integrity, delivery, security, and support for the BizTalk Framework and other key document formats.

In addition to BizTalk Server 2000, Microsoft, with industry partners, has led the way in innovation on the enabling technologies that are necessary for Internet–based business solutions, including BizTalk Framework 2.0, which is a platform–independent, Extensible Markup Language (XML) framework for application integration and electronic commerce. BizTalk Framework 2.0 is not a standard, but it builds on existing standards such as SOAP. SOAP is also a key technology in other members of the .NET product line, such as Microsoft Visual Studio .NET BizTalk Framework 2.0 provides the basis for interoperable, reliable messaging for BizTalk Server 2000.

Design Goals

The following design goals were under consideration as Microsoft created the BizTalk Server product:

- Provides an interface for managing application–to–application data transformation.
- Provides tools and services to manage the exchange of data between business partners.
- Provides control over long–running transactions and business processes.
- Guarantees the reliable delivery of documents and messages.

Features

Microsoft BizTalk Server 2000 provides tools and services that allow you to create executable applications for controlling your business processes and the exchange of data between trading partners and applications within your business. The following features make the mentioned advantages possible:

- BizTalk Orchestration Designer is a Microsoft Vision 2000–based design tool that enables you to create business process drawings that can be compiled and run as XLANG

schedules. XLANG is an XML−based language. An XLANG schedule describes the business process and the binding of that process to application services. You can use the BizTalk Orchestration Designer to create drawings that describe long−running, loosely coupled, executable business processes. Typically, these drawings describe the way interactions and procedures are performed during the completion of a specified process such as a purchase order request. Often these business processes are not constrained by time limits. Also, the steps in a business process are loosely coupled. The description of the business process is separate from the implementation logic and sequencing used to perform the process.

- Microsoft BizTalk Server 2000 provides two methods for configuring BizTalk Messaging Services to manage the exchange of documents between business partners and applications within a business. You can use BizTalk Messaging Manager, which is a graphical user interface, or directly access the BizTalk Messaging Configuration object model. BizTalk Server 2000 provides reliable document delivery by using configurable BizTalk Messaging Services properties. These properties include setting service windows for sending documents, sending or receiving receipts, setting the number of retries, and setting the time between retries.
- BizTalk Server provides tools with which you can define the structure of a document and map data from one format to another. These tools are based on XML and provide the essential data translation necessary for an application−integration server.
- BizTalk Server 2000 provides receive functions that enable the server to monitor documents posted at specified locations. BizTalk Server 2000 supports both file and message queuing receive functions. You can configure functions to occur when documents are posted to one of the receipt locations you have configured.
- BizTalk Server 2000 provides data validation by verifying each instance of a document against a specification. If a document does not adhere to the specification rules, that document is placed into a suspended queue that you can use for further analysis.
- BizTalk Server 2000 supports encryption and digital signatures. Public−key encryption technology is supported for all documents that are transmitted by using BizTalk Server 2000 transport services. BizTalk Server 2000 also supports decryption and signature verification for the documents it receives.

Internet Security and Acceleration Server

Internet Security and Acceleration Server 2000 provides secure, fast, and manageable Internet connectivity. ISA Server integrates an extensible multi−layer enterprise firewall and a scalable high−performance Web cache. It builds on the .NET Server security model and directory for policy−based security, acceleration, and management of internetworking.

Design Goals

In creating ISA Server, Microsoft wanted to ensure that it provided the quickest possible Internet connection while providing an industry−standard firewall. The following goals had to be met as the product was created:

- Provides secure Internet connectivity. The internal resources have to be protected against intrusion.
- Provides fast Internet access.
- Provides centralized management of both security and acceleration features.
- Provides centralized management of multiple servers for larger−scale implementations.

Features

System administrators should be aware of all the new features provided with ISA Server. It is a significant upgrade from the previous Proxy Server products. Most administrators are concerned about the security of the network and want to implement packet filtering and intrusion–detection features to help stop hackers and control network traffic flow. This product is a firewall solution as well as an Internet access solution. However, it can also be a deterrent to Web applications. XML and SOAP were created in large part because of firewalls. These newer Internet protocols allow for data to be transferred via port 80 (similar to HTTP traffic). This allows the data to move more easily through the security infrastructure you have created. Web developers and system administrators must work together to understand how to use ISA Server to protect the network while allowing the application flexibility required by the current business processes. The following features should be researched when you are considering ISA Server for your network:

- **Packet filtering.** ISA Server allows network packets to be filtered by Internet Protocol (IP) address, port number, or protocol.
- **Intrusion detection.** ISA Server includes several options for intrusion detection. It checks for a variety of common attacks and notifies you of the problems that occur.
- **Caching.** ISA Server provides several levels of caching. This feature stores a copy of the item retrieved from the Internet to supply it locally. This can significantly increase Internet access to common sites for your users.
- **Server publishing.** You can publish internal resources to the Web. Web users interact with the ISA Server, which then retrieves the data from the internal server. This can be a very secure option for making data available to the Internet.

Review Questions

1. What are the common features among the .NET Enterprise Servers?
2. What is the role of XML in the .NET Enterprise Servers?
3. What are the products that make up the .NET Server platform?
4. What are the features of SQL Server that facilitate enterprise development?
5. What is the purpose of the Web Storage System of Exchange Server?
6. What is the difference between Commerce Server and Content Management Server?
7. What features of SharePoint Portal Server help create user–friendly Web sites?
8. What is the purpose of Application Center Server?
9. What is the BizTalk Orchestration Designer?

Chapter 4: .NET Services for the Development Environment

Overview

The .NET vision was created with cross–platform integration both between systems and between organizations in mind. This vision is enabled by Windows .NET Server native support for XML Web services implemented in a manner that is consistent with the underlying industry and Internet standards. This system permits applications to incorporate loosely coupled Internet applications and services across disparate environments to integrate applications and services seamlessly.

While XML and Web services provide the foundation for application integration, additional .NET services focus on development with the .NET platform. This chapter discusses the services that directly affect the developer on a day–to–day basis. Chapter 5, "XML Integration with .NET Servers," introduces XML and its underlying technologies; Web services are addressed in more detail in Chapter 15, "Web Services with .NET"

The Windows .NET server application environment helps improve developers' productivity by providing a complete set of application tools and services. By integrating the .NET Framework into the Windows .NET server application development environment, developers are free to write applications without spending an inordinate amount of time developing the "plumbing" code. As a result, developers are more productive, can predict development timelines better, and can deploy an application faster than is possible in other development platforms. The .NET vision provides a framework that makes application development scalable, fast, and easy.

To better understand the services discussed in this chapter the reader should be comfortable with the .NET Framework. The .NET Framework is described in more detail in Chapter 1, "Introduction to .NET Framework." Figure 4.1 identifies the three layers of the .NET Framework and can serve as a quick refresher. At the foundation is the common language runtime layer, which manages the execution of .NET code. The base classes layer builds on this foundation with an extensible set of class libraries, followed by the program and user interface layer.
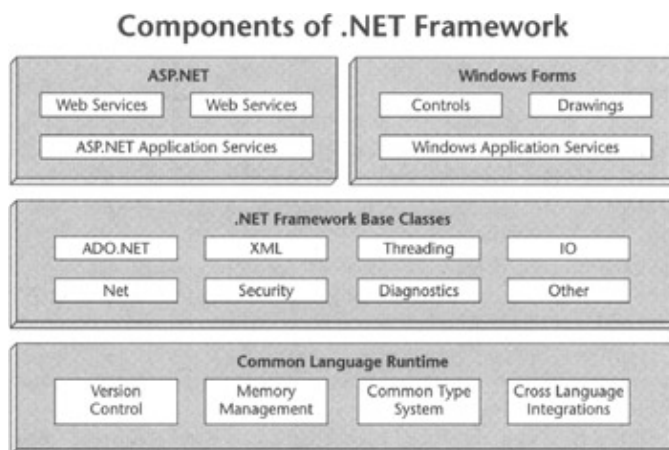
Figure 4.1: The .NET Framework has three layers that make applications easy to write and quick to deploy.

This chapter is organized in accordance with the layers of the .NET Framework. The services that are introduced come from the layers as they are outlined in the following list. Each of these items has a section later in the chapter that provides more depth on each layer's corresponding services.

- **Common language runtime services.** The common language runtime services include multilanguage support, metadata and versioning, and garbage collection.
- **Base class services.** The base class services include ADO.NET, .NET security, and remoting services.
- **Program and user interfaces.** The program and user interfaces include ASP.NET and Web services. Web services are introduced in this chapter. Chapter 15, "Web Services with .NET," provides additional detail on that topic.

Common Language Runtime Services

The .NET Framework provides a runtime environment called the common language runtime that manages the execution of code and provides services that make the development process easier. Compilers and tools expose the runtime's functionality and enable developers to write code in a way that benefits from the managed execution environment. Any code that is developed with a .NET−compliant language compiler (Visual Basic .NET, C#, and others) and targets the common language runtime is called managed code. Managed code takes advantage of runtime features such as cross−language integration, cross−language exception handling, enhanced security, versioning and deployment support, a simplified model for component interaction, and debugging and profiling services. The following sections identify the services that can be used to enhance application development and take advantage of the common language runtime services. The services described here include multiple language support, metadata and versioning, and garbage collection (automatic memory management).

Multiple Language Support

The .NET Framework provides for cross−language integration, which allows a developer to extend one programming language's components to another language by means of cross−language inheritance, debugging, and error handling. This allows the user to extend the functionality of a single language by incorporating it into another language to speed the development process.

The common language runtime makes it easy to design components and applications whose objects interact across languages: Objects written in different languages can interact with one another. The behaviors of those objects can be tightly integrated regardless of the development language. For example, it is possible to define a class in one language then use another language to make a method call on it or even derive an entirely new class from the original. One also can pass an instance of a class to a method on a class written in a different language. Cross−language integration is possible because the language compilers and tools that target the common language runtime use a common type system defined by the common language runtime. The compilers and tools also follow the runtime's rules for defining new types as well as for creating, using, persisting, and binding to types.

To take full advantage of cross−language object interaction, objects must expose only the features that are common to all the languages with which they interoperate. For this reason, a set of language features has been defined, called the Common Language Specification (CLS). The CLS includes the basic language features needed by most applications. The CLS rules were created to define a subset of the common type system. With that relationship, all the rules that apply to the common type system apply to the CLS except in cases where stricter rules have been defined in the

CLS. The CLS helps enhance and ensure language interoperability by defining a set of features that a developer can be sure are available in a wide variety of languages, such as Visual Basic .NET, C#, and Visual C++. The CLS also establishes requirements for CLS compliance. The compliance rules help you determine whether your managed code conforms to the CLS. These rules also help you determine the extent to which a specific tool supports the development of managed code that uses CLS features.

If the component that is created uses only CLS features in the Application Programming Interface (API) that it exposes to other code (including derived classes), that component will be accessible from any programming language that supports the CLS. The components that adhere to the CLS rules and use only the features included in the CLS then are identified as CLS−compliant components.

The CLS was designed to be large enough to include the language constructs commonly needed by developers. At the same time it is small enough to enable most languages to support it. Additionally, any language feature that was not able to verify the type safety of code quickly was excluded. This was done so that all CLS−compliant languages can produce verifiable code. It is still up to the language to do this, and so this is not a guarantee that all languages will have verifiable code.

Metadata and Versioning

Metadata is the supporting information describing a program that is stored either in a common language runtime portable executable file (PE file) or in memory. Metadata is stored in a binary format. When you compile code into a PE file, metadata is inserted into one portion of the file, while the code is converted to Microsoft Intermediate Language (MSIL) and inserted into another portion of the file. Every type and member defined and referenced in a module or assembly is described within metadata. The binary format keeps this process small and manageable. An assembly is a single deployable unit or Visual Studio .NET project. When the code of the application is executed, the runtime loads metadata into memory and references it to discover information about the code's classes, members, inheritance, and so on.

As part of the stored metadata, managed components carry information about the components and resources on which they were built. The runtime uses this information to ensure that the component or application has the appropriate versions of everything it needs. This decreases the likelihood of failure resulting from an unmet dependency. Registration information and state data traditionally were stored in the operating system registry. This system was changed to increase the readability and accessibility of the information required by the application. This also results in cleaner installations and uninstalls of the application. The information about the types that are defined (and their dependencies) is stored with the code as metadata, making the tasks of component replication and removal much less complicated.

Metadata describes every type and member defined in your code by storing the information described in the following list:

- **Description of the assembly.** The identity of the assembly and the types (classes) it exposes. Metadata also defines the other assemblies it depends on as well as the security permissions necessary to invoke it.
- **Description of types.** The names of all the types defined and used, including base classes and implemented interfaces. Additionally, all members of the type are described in a manner similar that used in pre−.NET type libraries.

Metadata is the key to a simpler programming model; it eliminates the need for Interface Definition

Language (IDL) files, header files, and any other external methods of component reference. Metadata allows .NET languages to describe themselves automatically in a manner that is not seen by the developer and the user. The benefits of metadata include the following:

- **Self−describing files.** Common language runtime modules and assemblies are self−describing. A module's metadata contains everything needed to interact with another module. Runtime modules and assemblies do not require registration with the operating system. As a result, the descriptions used by the runtime always reflect the actual code in the developer's compiled file, and this increases application reliability and versioning control.
- **Language interoperability and easier component−based design.** Metadata provides all the information about compiled code required for you to inherit a class from a PE file written in a different language. You can create an instance of any class written in any managed language (any language that targets the common language runtime) without worrying about explicit marshaling or using custom interoperability code.
- **Definition and storage of attributes.** The .NET Framework allows you to declare specific kinds of metadata, called attributes, in your compiled file. Attributes can be found throughout the .NET Framework and are used to control in more detail the way a program behaves at runtime. Additionally, you can emit your own custom metadata into .NET Framework files through user−defined custom attributes.

Garbage Collection

The common language runtime is referred to as a garbage−collected environment. Garbage collection is a feature that cleans up the object creation and reference memory usage of an application. Garbage collection frees the application from the task of explicitly destroying objects when they are no longer required. This significantly reduces common programming errors and decreases the likelihood of memory leaks in an application.

The runtime automatically manages object layout and all references to the objects, releasing them when they are no longer in use. Objects whose lifetimes are managed by the runtime in this way are called managed data. Garbage collection is a mechanism that allows the system to detect when an object can no longer be accessed. It then automatically releases the memory used by that object (as well as calling a cleanup routine, called a "finalizer," that may be written by the developer). Some garbage collectors, such as the one used by .NET, compact memory and therefore decrease a program's working set. This frees up memory for it and other applications.

The garbage collector provides the following advantages to programmers:

- No more worrying about deallocating memory
- No need to reference counting objects regardless of the sophistication of the data structure
- No need to worry about memory leaks

If you typically deallocate system resources (file handles, locks, and so forth) in the same block of code that releases the memory for an object, you will have to make some changes in your coding style to take advantage of garbage collection. To facilitate garbage collection, you should provide a method that releases the system resources programmatically and let the garbage collector release the memory when it compacts the working set. All languages that target the runtime allow you to allocate class objects from the garbage−collected heap. This allows the objects to be allocated quickly and avoids the need for programmers to know how and when to free an object.

The common language runtime also provides an option referred to as ValueTypes. ValueTypes are similar to classes except that ValueType objects are allocated on the runtime stack rather than on

the heap. This results in objects that are reclaimed automatically when code exits a procedure in which the object is defined.

Base Class Services

The next layer in the .NET Framework containing services for application developers is the base class layer. While numerous services exposed in class library API are defined here, data access, security, and remoting services are some of the most significant services for developers.

Accessing Data with ADO.NET

Accessing data is generally a requirement for programmers for both standalone and Web–based applications. Microsoft's ADO.NET technology offers a solution to many of the problems associated with data access.

ADO.NET represents a step up from its predecessor, Microsoft ActiveX Data Objects (ADO). It is a programming model for developing distributed data–sharing applications. ADO.NET offers several advantages over previous versions of ADO as well as other data access technologies, including enhanced interoperability, maintainability, programmability, and performance.

The following sections provide an overview of ADO.NET, a discussion of its design goals, and an overview of the ADO.NET architecture.

Overview of ADO.NET

ADO.NET provides consistent access to data sources such as Microsoft SQL Server as well as data sources exposed via OLE DB and XML. Data–sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, manipulate, and update data.

ADO.NET cleanly factors data access from data manipulation into discrete components that can be used separately or in tandem. ADO.NET includes .NET data providers for connecting to a database, executing commands, and retrieving results. Those results are processed directly or placed in an ADO.NET DataSet object to be exposed to the user in an ad hoc manner, combined with data from multiple sources, or remoted between tiers. The ADO.NET DataSet object also can be used independently of a .NET data provider to manage data local to the application or sourced from XML.

Design Goals

As application development has evolved, new applications have become loosely coupled on the basis of the Web application model. More and more of today's applications use XML to encode data that will be passed over network connections. Web applications use HTTP as the protocol for communication between tiers and consequently are responsible for maintaining state between requests. This is quite different from the old two–tier client/server model, in which connections were held open for the duration of the application and application state was maintained constantly.

To facilitate its .NET vision, Microsoft recognized that a new data access programming model was needed, one built on the .NET Framework. The .NET Framework would ensure that all components would share a common type system, design patterns, and naming conventions.

ADO.NET was designed to meet the needs of this new programming model. Its design focuses on a disconnected data paradigm with tight XML integration. Additionally, it enables data from

heterogeneous data sources to be combined and analyzed. The following list identifies the design goals of ADO.NET:

- **Leverage current ADO knowledge.** Microsoft's design for ADO.NET addresses many of the requirements of today's application development model. The ADO.NET programming model maintains as much compatibility with ADO as possible so that current ADO developers do not have to start from scratch in learning a new data access technology. ADO.NET is an intrinsic part of the .NET Framework without seeming foreign to ADO programmers. ADO.NET coexists with ADO. While most new .NET applications will be written using ADO.NET, ADO remains available to .NET programmers through .NET COM interoperability services.
- **Support the *n*–tier programming model.** ADO.NET provides first–class support for the disconnected *n*–tier programming environment for which many new applications are written. The concept of working with a disconnected set of data has become a focal point in the programming model. The ADO.NET solution for *n*–tier programming is the DataSet.
- **Support XML.** XML and data access are intimately intertwined: XML is all about defining data, and data access is becoming all about XML. The .NET Framework not only supports Web standards but uses them as its foundation. XML support is built into ADO.NET at a fundamental level. The XML class framework in .NET and ADO.NET are part of the same architecture and are tightly integrated. It is no longer necessary to choose between the data access sets of services and their XML counterparts. Both are designed to be interoperable.

ADO.NET Architecture

Data processing traditionally relied primarily on a connection–based two–tier model. As data processing increasingly uses multi–tier architectures, programmers are switching to a disconnected approach to provide better scalability for their applications. The following sections explain the parts of the ADO.NET architecture that facilitate scalability and support for the development of multi–tier applications. The core of the architecture allows for scalability to easily support any type of application the developer would like to write. Figure 4.2 identifies the core components of the ADO.NET architecture.
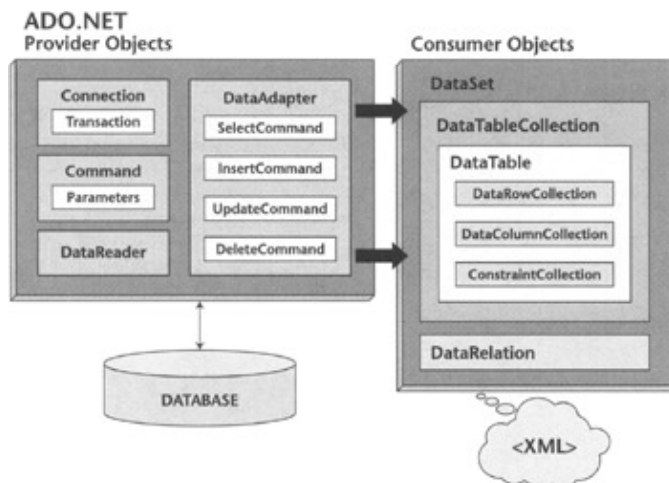
Figure 4.2: The ADO.NET architecture is easy to implement, flexible for any type of application, and scalable to meet all data requirements.
XML and ADO.NET

ADO.NET leverages the power of XML to provide disconnected data access. It was designed in concert with the .NET Framework; as a result, both are components of a single architecture.

ADO.NET and the .NET Framework converge in the DataSet object. The DataSet can be populated with data from a database or an XML source, whether that source is a file or a data XML stream. The DataSet can be written as a valid XML document that conforms to its XML schema. Because the native format of the DataSet is XML, it is an excellent medium for moving data between tiers in a distributed environment, making it an ideal choice for transmitting data to and from a Web service.

ADO.NET Components

The ADO.NET components have been designed to separate data access from data manipulation. There are two central components of ADO.NET that accomplish this: the DataSet, which is an ADO.NET consumer object, and the .NET data provider, which is a set of components that includes the connection, command, DataReader, and DataAdapter objects, as depicted in Figure 4.2. Although data access and data manipulation may be performed at the same logical time, it often is desirable to have control over each process.

The DataSet is the core component of the disconnected architecture of ADO.NET. It is designed for data access independent of any data source. As a result, it can be used with multiple and differing data sources or XML data or used to manage data local to the application. The DataSet contains a collection of one or more DataTable objects made up of rows and columns of data as well as primary key, foreign key, constraint, and relation information about the data in the DataTable objects.

The other core element of the ADO.NET architecture is the .NET data provider, whose components are designed explicitly for data manipulation and fast, forward–only, read–only access to data. The connection object provides connectivity to a data source. The command object enables access to database commands to return data, modify data, run stored procedures, and send or retrieve parameter information. The DataReader provides a high–performance stream of data from the data source. Finally, the DataAdapter provides the bridge between the DataSet object and the data source. The DataAdapter uses command objects to execute SQL commands at the data source to load the DataSet with data and reconcile changes made to the data in the DataSet with the data source.

.NET data providers can be written for any data source. The .NET Framework ships with two .NET data providers: the SQL Server .NET Data Provider and the OLE DB .NET Data Provider.

.NET Security

The Microsoft .NET Framework offers code access security and role–based security to address security concerns in distributed applications. These security mechanisms use a simple, consistent model, and so developers familiar with code access security can easily use role–based security and those familiar with role–base security can use code access security. Both code access security and role–based security are implemented through the use of a common infrastructure supplied by the common language runtime. The following sections describe code access security and role–based security.

Introduction to Code Access Security

Today's highly connected computer systems frequently are exposed to code originating from various and possibly unknown sources. Code can be attached to email, contained in documents, and downloaded over the Internet. Many computer users have experienced the effects of malicious mobile code, including viruses and worms, which can damage or destroy data and cost time and money.

Most common security mechanisms give rights to users on the basis of their logon credentials (usually a password) and restrict the resources (often directories and files) a user is allowed to access. However, this approach fails to address the following issues:

- Users obtain code from many sources, some of which may be unreliable.
- Imperfect code can be exploited by malicious code.
- Code sometimes does things the user does not know it will do. As a result, computer systems can be damaged and private data can be leaked when cautious and trustworthy users run malicious or buggy software.

To help protect computer systems from malicious code, allow code from unknown origins to run safely, and prevent trusted code from intentionally or accidentally compromising security, the .NET Framework provides a security mechanism called code access security. Code access security allows code to be trusted to varying degrees, depending on where the code originates and other aspects of the code's identity. Code access security also enforces these varying levels of trust on code, minimizing the amount of code that must be fully trusted in order to run. Using code access security can reduce the likelihood that code can be misused by malicious or buggy code. It can reduce your liability because you can specify the set of operations the code should be allowed to perform as well as the operations the code should never be allowed to perform. Code access security also can minimize the damage that can result from security vulnerabilities in code.

Code access security is a mechanism that protects resources and operations from being accessed by untrusted code. In the .NET Framework code access security performs the following functions:

- Defines permissions and permission sets that represent the right to access various system resources.
- Enables administrators to configure security policy, which associates sets of permissions with code groups.
- Allows code to request the permissions it requires in order to run, the permissions that it would be useful to have, and the permissions the code must never have.
- Grants permissions to each assembly that is loaded, based on the permissions requested by the code and the operations permitted by security policy.
- Enables code to demand that its callers have specific permissions.
- Enables code to demand that its callers possess a digital signature, allowing only callers from a particular organization or site to call the protected code.
- Enforces restrictions on code at runtime by comparing the granted permissions of every caller on the call stack with the permissions that callers must have.

To determine whether code is authorized to access a resource or perform an operation, the runtime's security system walks the call stack, comparing the granted permissions of each caller with the permission being requested. If any caller in the call stack does not have the permission that was requested, a security exception is thrown and access is denied. The stack walk is designed to prevent luring attacks in which less trusted code calls highly trusted code and uses it to perform unauthorized actions.

While a request for permission at runtime affects performance, it is essential to protect code from luring attacks by less trusted code. If performance optimizations are required, code can be written so that fewer stack walks are performed; however, the programmer must take responsibility for not exposing a security weakness whenever this is done. Figure 4.3 illustrates this process through a stack walk that results when a method in Assembly A3 requests that its callers have permission P.

Security Stack Walk
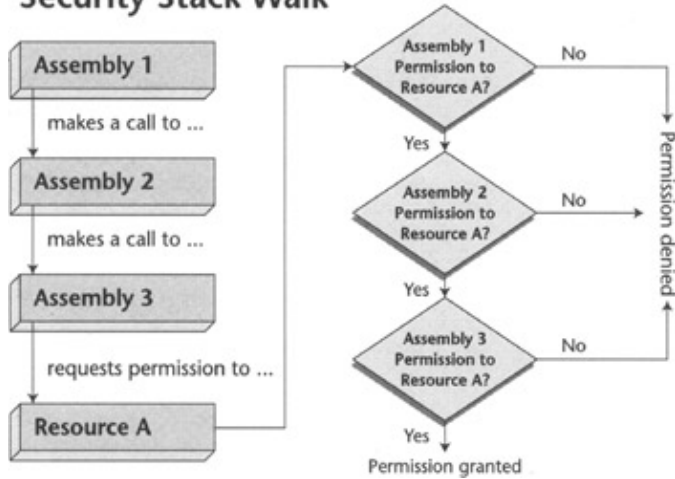
Figure 4.3: Security stack walk example.
Code Access Security Basics

Every application that targets the common language runtime must interact with the runtime's security system. When an application executes, it is automatically evaluated and given a set of permissions by the runtime. Depending on the permissions the application receives, it either runs properly or generates a security exception. The local security settings on a particular machine ultimately decide which permissions code receives. Because these settings can change from machine to machine, one can never be sure that code will receive sufficient permissions to run. This is in contrast to the world of unmanaged development, in which developers do not have to worry about their code's permission to run.

Every developer must be familiar with the following code access security concepts to write effective applications targeting the common language runtime:

- **Writing type–safe code.** To enable code to benefit from code access security, application and component developers must use a compiler that generates verifiably type–safe code.
- **Imperative and declarative syntax.** Interaction with the runtime security system is performed by using imperative and declarative security calls. Declarative calls are performed by using attributes, and imperative calls are performed by using new instances of classes within the code. Some calls can be performed only imperatively, and others can be performed only declaratively. Some calls can be performed in either manner.
- **Requesting permissions for code.** Requests are applied to the assembly scope, where your code informs the runtime about permissions that it either needs to run or specifically does not want. Security requests are evaluated by the runtime when your code is loaded into memory. Requests cannot influence the runtime to give your code more permission than the runtime would have given if the request had not been made. However, requests are the way your code informs the runtime about the permissions it requires to run.
- **Using secure class libraries.** Code access security allows your class libraries to specify the permissions they require in order to be accessed. You should be aware of the permissions required to access any library your code uses and make appropriate requests in that code.

Introduction to Role–Based Security

Business applications often provide access to data or resources on the basis of credentials supplied by the user. Typically, such applications check the role of a user and provide access to resources on the basis of that role. The common language runtime provides support for role–based

authorization that is based on a Windows account or a custom identity.

Roles often are used in financial or business applications to enforce business rules. For example, an application might impose limits on the size of the transaction being processed, depending on whether the user making the request is a member of a specified role. Clerks might have authorization to process transactions that are less than a specified threshold, supervisors might have a higher limit, and vice presidents might have a still higher limit or no limit at all. Role–based security also can be used when an application requires multiple approvals to complete an action. An example might be a purchasing system in which any employee can generate a purchase request but only a purchasing agent can convert that request into a purchase order that can be sent to a supplier.

.NET Framework role–based security supports authorization by making information about the principal, which is constructed from an associated identity, available to the current thread. The identity (and the principal it helps define) can be based on a Windows account or be a custom identity unrelated to a Windows account. .NET applications can make authorization decisions on the basis of the principal's identity or role membership or both.

A role is a named set of principals that have the same privileges with respect to security, such as a teller or a manager. A principal can be a member of one or more roles. Therefore, applications can use role membership to determine whether a principal is authorized to perform a requested action.

To provide ease of use and consistency with code access security, .NET Framework role–based security provides PrincipalPermission objects that enable the common language runtime to perform authorization in a way that is similar to code access security checks. The PrincipalPermission class represents the identity or role that the principal must match and is compatible with security checks. You also can access a principal's identity information directly and perform role and identity checks in your code when needed.

The .NET Framework provides role–based security support that is flexible and extensible enough to meet the needs of a wide spectrum of applications. You can choose to interoperate with existing authentication infrastructures such as COM+ applications or create a custom authentication system. Role–based security is particularly well suited for use in ASP.NET Web applications, which are processed primarily on the server. However, .NET Framework role–based security can be used on either the client or the server.

Remoting Services

Establishing communication between objects that run in different processes, whether on the same computer or on computers in different countries, is a common goal in the development of distributed applications. Traditionally, this required in–depth knowledge not only of the objects on either end of the communication stream but also of a host of lower–level protocols, application programming interfaces, and configuration tools or files. In short, it was a complex task that demanded substantial concentration and experience.

The .NET Framework makes several communication methods available to accomplish this task quickly and easily whether or not a developer knows much about protocols and encodings. As a result, whether you need to develop a Web application quickly or spend more time building an enterprisewide application involving many computers or operating systems and using multiple protocols, the .NET Framework can support you.

Communicating across processes is still a complex task, but much of it is now handled by the .NET

Framework. The following sections outline the details of .NET remoting. The first a section provides an overview of .NET remoting, then a section outlines differences between copies and references, and finally a section helps explain the .NET remoting architecture.

.NET Remoting Overview

.NET remoting can be used to enable different applications to communicate with one another, whether those applications reside on the same computer, on different computers in the same local area network, or across the world in very different networks. This applies even though those computers may be running on different operating systems.

The remoting infrastructure is an abstract approach to interprocess communication. Much of the system functions without drawing attention to itself. For example, objects that can be passed by value, or copied, are passed automatically between applications in different application domains or on different computers. You need only mark your custom classes as serializable (with the ability to persist an object to a file) to make this work.

The real strength of the remoting system, however, resides in its ability to enable communication between objects in different application domains or processes using different protocols, serialization formats, object lifetime schemes, and modes of object creation. In addition, if there is a need to intervene in almost any stage of the communication process for any reason, remoting makes this possible.

Whether you have implemented a number of distributed applications or are simply interested in moving components to other computers to increase the scalability of your program, it is easiest to understand the remoting system by thinking of it as a system of interprocess communication with some default implementations that easily handle most scenarios. The following discussion begins with the basics of interprocess communication using remoting.

Copies versus References

Cross−process communication requires a server object whose functionality is provided to callers outside its process. A client makes calls to the server, and a transportation mechanism ferries the calls from one end to the other. The addresses of server methods are logical and function properly in one process but do not function in a different client process. To alleviate this problem, an entire copy of the server object can be moved to the client process where the copy's methods can be invoked directly.

Many objects, however, cannot or should not be copied and moved to another process for execution. Large objects with many methods are poor choices for copying, or passing by value, to other processes. Usually, a client needs only the information returned by one method or a few methods on the server object. Copying the entire server object, including what could be vast amounts of internal information or executable structures unrelated to the client's needs, would be a waste of bandwidth as well as of client memory and processing time. In addition, many objects expose public functionality but require private data for internal execution. Copying these objects could enable malicious clients to examine internal data, creating the potential for security problems. Finally, some objects use data that cannot be copied in an understandable way. A FileInfo object, for example, contains a reference to an operating system file, which has a unique address in the server process's memory. You can copy this address, but it will never make sense in another process.

In these situations the server process should pass to the client process a reference to the server

object rather than a copy. Clients can use this reference to call the server object. These calls do not execute in the client process. Instead, the remoting system collects all the information about the call and sends it to the server process, where it is interpreted, the server object is located, and the call is made to the server object on the client object's behalf. The result of the call then is sent back to the client process to be returned to the client. Bandwidth is used for only the critical information: the call, the call arguments, and any return values or exceptions.

Remoting Architecture Simplified

Using object references to communicate between server objects and clients is the heart of remoting. The remoting architecture, however, presents to the programmer an even simpler procedure. If you configure the client properly, you need only create a new instance of the remote object by using a new constructor. The client receives a reference to the server object, and you then can call its methods as though the object were in your process rather than running on a separate computer. The remoting system uses proxy objects to create the impression that the server object is in the client's process. Proxies are stand–in objects that present themselves as an other object. When the client creates an instance of the remote type, the remoting infrastructure creates a proxy object that looks to the client exactly like the remote type. The client calls a method on that proxy, and the remoting system receives the call, routes it to the server process, invokes the server object, and returns the return value to the client proxy, which returns the result to the client.

Remote calls must be conveyed in some way between the client and the server process. If you were building a remoting system, you might start by learning network programming and a wide array of protocols and serialization format specifications. In the .NET remoting system, the combined underlying technologies required to open a network connection and use a particular protocol to send the bytes to the receiving application are represented as a transport channel.

A channel is a type that takes a stream of data, creates a package according to a particular network protocol, and sends the package to another computer. Some channels can only receive information, others can only send information, and still others, such as the default TcpChannel and HttpChannel classes, can be used in either way.

Although the server process knows everything about each unique type, the client knows only that it wants a reference to an object in another application domain, perhaps on another computer. From the world outside the server application domain, the object is located by a Uniform Resource Locator (URL). The URLs that represent unique types to the outside world are activation URLs, which ensure that your remote call is made to the proper type.

If both sides of the relationship are configured properly, a client merely creates a new instance of the server class. The remoting system creates a proxy object that represents the class and returns to the client object a reference to the proxy. When a client calls a method, the remoting infrastructure fields the call, checks the type information, and sends the call over the channel to the server process. A listening channel picks up the request and forwards it to the server remoting system, which locates (or creates, if necessary) and calls the requested object. The process then is reversed as the server remoting system bundles the response into a message that the server channel sends to the client channel. Finally, the client remoting system returns the result of the call to the client object through the proxy.

Very little actual code is required to make this work, but some thought should be given to the design and the configuration of the relationship. The code can be absolutely correct yet fail because a URL or port number is incorrect.

Program and User Interfaces

The final .NET Framework layer containing .NET services for developers is the program and user interface layer. The two principal services in this layer form the cornerstone of Microsoft's .NET vision: ASP.NET and Web services. This section focuses on the ASP.NET services. Web services are addressed more completely in Chapter 15, "Web Services with .NET."

Introduction to ASP.NET

ASP.NET is more than the next version of Active Server Pages (ASP); it is a unified Web development platform that provides the services needed by developers to build enterprise–class Web applications. While ASP.NET is largely syntax–compatible with ASP, it also provides a new programming model and an infrastructure that enable a powerful new class of applications. ASP and ASP.NET can coexist. You can freely augment your existing ASP applications by incrementally adding ASP.NET functionality.

ASP.NET is a compiled .NET–based environment; you can author applications in any .NET–compatible language, including Visual Basic, C#, and JScript. Additionally, the entire .NET Framework is available to any ASP.NET application. Developers can easily access the benefits of these technologies, which include a managed common language runtime environment, type safety, and inheritance, among others.

ASP.NET has been designed to work seamlessly with HTML editors and other programming tools, including Microsoft Visual Studio .NET. Not only does this make Web development easier, it also provides all the benefits these tools have to offer, including a graphical user interface (GUI) that developers can use to drop server controls onto a Web page as well as fully integrated debugging support.

Developers can choose from or combine two programming models when creating an ASP.NET application:

- **Web Forms allow you to build powerful forms–based Web pages.** In building these pages, you can use ASP.NET server controls to create common user interface (UI) elements and program them for common tasks. These controls allow you to build up a Web Form rapidly out of reusable built–in or custom components, simplifying the code of a page.
- **Web services can access server functionality remotely.** Using services, businesses can expose programmatic interfaces to their data or business logic, which in turn can be obtained and manipulated by client and server applications. Web services enable the exchange of data in client/server or server/server scenarios by using standards such as HTTP and XML messaging to move data across firewalls. Web services are not tied to a particular component technology or object–calling convention. As a result, programs written in any language, using any component model, and running on any operating system can access Web services. For more information on Web services, refer to Chapter 15, "Web Services with .NET."

Web Forms

Web Forms are an ASP.NET technology that is used to create programmable Web pages. They can present information, using any markup language, to the user in any browser and use code on the server to implement application logic. The following list identifies the considerations that should precede the development of Web Forms:

- Can run on any browser and automatically render the correct, browser–compliant HTML for features such as styles and layout. Alternatively, you can design your Web Form to run on a specific browser, such as Microsoft Internet Explorer 5, and take advantage of the features of a rich browser client.
- Can be programmed in any .NET–supported language, such as Visual Basic, C#, and JScript.NET.
- Are built on the common language runtime and provide all the benefits of that technology, including a managed execution environment, type safety, inheritance, and dynamic compilation for improved performance.
- Support editing and powerful rapid application development (RAD) development tools such as Microsoft Visual Studio .NET for designing and programming forms.
- Support a rich set of controls that allow developers to encapsulate page logic into reusable components cleanly and declaratively handle page events.
- Allow for separation between code and content on a page, eliminating the "spaghetti code" often found in ASP pages.
- Provide a set of state management features that preserve the view state of a page between requests.
- Are extensible with user–created and third–party controls.

Components of Web Forms

Web Forms divide the Web applications user interface into two pieces: the visual component and the user interface logic. If you have worked with rapid application deployment tools such as Microsoft Visual Basic and Microsoft Visual C++, you will recognize this distinction between the visible portion of a form and the code that interacts with the form.

The user interface for Web Forms pages consists of a file containing markup and Web Forms–specific elements. This file is referred to as the page. The page works as a container for the text and controls you want to display. Using any HTML editor plus Web Forms server controls, you can lay out the form as you like. The page is a file with the extension .aspx.

User interface logic for the Web Form consists of code that you create to interact with the form. You can specify that the programming logic reside in the .aspx file or in a separate file (referred to as the "code–behind" file) written in Visual Basic, C#, or any other common language runtime–supported language. When you run the form, the code–behind class file runs and dynamically produces the output for your page.

What Web Forms Helps Accomplish

Web application programming presents challenges that typically do not arise in programming traditional client–based applications. Among the challenges are these:

- **Rich user interface.** A user interface with a large amount of content, a complex layout, and rich user interaction can be difficult and tedious to create and program using basic HTML facilities. It is especially hard to create a rich user interface for applications that are likely to run on many different browsers.
- **Separation of client and server.** In a Web application the client (browser) and the server are different programs that often run on different computers and even on different operating systems. Consequently, the two halves of the application share little information; they can communicate but typically only exchange small chunks of simple information.
- **Stateless execution.** When a Web server receives a request for a page, it finds the page, processes it, sends it to the browser, and then effectively discards all page information. If the user requests the same page again, the server repeats the entire sequence, reprocessing

the page from scratch. Put another way, servers have no memory of the pages they have processed. Therefore, if an application needs to maintain information about a page, this becomes a problem that has to be solved in application code.

- **Unknown client capabilities.** In many cases Web applications are accessible to many users that employ different browsers. Each of these browsers has different capabilities, making it difficult to create an application that will run equally well on all of them.
- **Data access.** Reading from and writing to a data source in traditional Web applications can be complicated and resource–intensive.

Meeting these challenges for Web applications can require substantial time and effort. Web Forms address these challenges. The following features can overcome the challenges related to developing Web applications:

- **Browser–independent applications.** Web Forms provides a framework for creating all application logic on the server, eliminating the need to code explicitly for differences in browsers. However, it still allows you to take advantage of browser–specific features automatically to provide improved performance and a richer client experience.
- **Event–based programming model.** Web Forms brings to Web applications the model of writing event–handling methods for events that occur in either the client or the server. The Web Forms framework abstracts this model in such a way that the underlying mechanism of capturing an event on the client, transmitting it to the server, and calling the appropriate handler is automatic and invisible. The result is a clear, easily written code structure.
- **Abstract, intuitive, consistent object model.** The .NET Framework presents an object model that allows you to think of forms as a unit, not as separate client and server pieces. In the Web Forms model, one can program the form in a much more intuitive way than can be done in traditional Web applications, including the ability to set properties for form elements and respond to events. In addition, Web Forms controls are an abstraction from the physical contents of an HTML page and from the direct interaction between browser and server. In general, you can use Web Forms controls the way you might work with controls in a client application and not have to think about how to create the HTML to present and process the controls and their contents.
- **State management.** The .NET Framework automatically handles the task of maintaining the state of a form and its controls and provides you with explicit ways to maintain the state of application–specific information. This is accomplished without the heavy use of server resources and without sending cookies to the browser, two traditional means for storing state.
- **Scalable server performance.** The .NET Framework allows you to scale an application from one computer with a single processor to a multicomputer Web farm cleanly and without complicated changes to the application's logic.

ASP.NET not only takes advantage of the performance enhancements in the .NET Framework and runtime, it also offers significant performance improvements over ASP and other Web development platforms. All ASP.NET code is compiled rather than interpreted, and this allows early binding, strong typing, and just–in–time (JIT) compiling to native code, to name only a few of the benefits.

ASP.NET is also easily factorable, meaning that developers can remove modules (a session module, for instance) that are not relevant to the application they are developing. ASP.NET also provides extensive caching services, both built–in and caching APIs. ASP.NET also ships with performance counters that developers and system administrators can monitor to test new applications and gather metrics on existing ones. The following list identifies some additional advantages of using ASP.NET:

- ASP.NET configuration settings are stored in XML−based files, which are human−readable and human−writable. Each of your applications can have a distinct configuration file, and you can extend the configuration scheme to suit your requirements.
- ASP.NET provides easy−to−use application and session state facilities that are familiar to ASP developers and are readily compatible with all other .NET Framework APIs.
- The .NET Framework and ASP.NET provide default authorization and authentication schemes for Web applications. You can easily remove, add to, or replace these schemes, depending on the needs of the application.
- Accessing databases from ASP.NET applications is a frequently used technique for displaying data to Web site visitors. ASP.NET makes it easier than ever to access databases for this purpose and makes it possible to manage the data in the database.
- ASP.NET provides a simple framework that enables Web developers to write logic that runs at the application level. Developers can write this code in either the global.asax text file or in a compiled class deployed as an assembly. This logic can include application−level events, but developers can easily extend this framework to suit the needs of their Web application. ASP application code written in the global.asa file is completely supported in ASP.NET. You can simply rename global.asa to global.asax when you are upgrading from ASP.
- ASP.NET offers complete syntax and processing compatibility with ASP applications. Developers simply have to change file extensions from .asp to .aspx to migrate their files to the ASP.NET page framework. They also can add ASP.NET functionality to their applications with ease, sometimes by simply adding just a few lines of code to their ASP files.

Review Questions

1. What are the three layers of the .NET Framework?
2. What is the advantage of multilanguage support?
3. What is the purpose of the garbage collector?
4. What are the services associated with the base class services layer of the .NET Framework?
5. What is the difference between code access security and role−based security?
6. What is the purpose of Web services and Web Forms?
7. What is the role of ASP.NET?
8. What do Web Forms help accomplish?

Chapter 5: XML Integration with .NET Servers

Many people wonder what XML is good for and how they can use it in their applications. The press, supported by numerous surveys and studies, has suggested that one cannot sit on the sidelines and put off using XML in enterprise applications. In fact, the foundation of the .NET Framework and the .NET Servers is their use of XML as the communication medium.

This chapter discusses the compelling need for a technology such as XML and the vital role it will play in enterprise and Web applications. It then introduces the underlying XML technologies and explains how they come together to deliver on its much–publicized promise.

Why XML?

Although the applications for XML are limited only by a user's imagination, there are fundamental uses of XML that are core to .NET Framework applications that are deployed on .NET Enterprise Servers. Before XML and its related technologies, true enterprise application development was hampered by the following:

- The lack of a standard messaging format both within and outside an organization. HTML is great for humans with their ubiquitous browsers, but what about business applications?
- The inability to identify data elements once they are presented in a Web application. Once data are presented in a Web application, the data no longer are directly accessible by the client.
- The inability to directly communicate through firewalls with "nontrusted" connections.
- The inability to use the services (methods) of components developed on other platforms. While many organizations attempt to standardize on a specific platform, they are not always successful. Additionally, e–commerce applications rarely can rely on a consistent operating platform or component development environment.

This section demonstrates how XML and its related technologies can be used to overcome these challenges. Figure 5.1 illustrates the most common uses of XML as they relate to Microsoft's .NET servers and the .NET Framework.
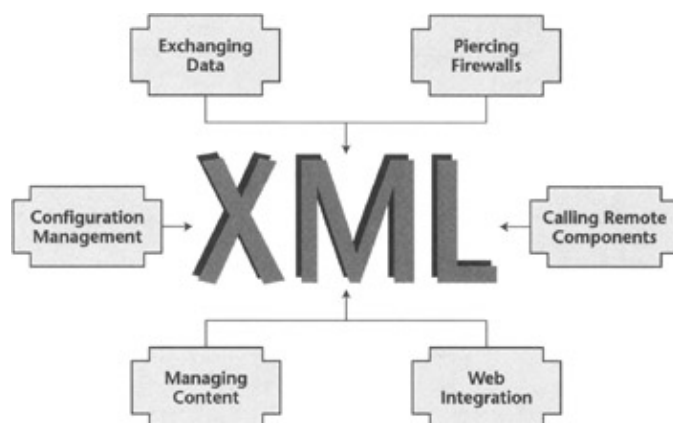
Figure 5.1: The common uses of XML technology, particularly as they relate to .NET servers and the .NET Framework.

Exchanging Data

Using a simple text–based markup mechanism, XML lets programs determine what the data are and how individual data items relate to each other. One can take almost any data storage requirement and see how it could be improved with XML. For example, many people want to keep lists of contacts with email addresses and phone numbers. Although this type of list seems simple, it can quickly become quite complex. For example, some of your contacts may have several phone numbers: one for work, one for home, and perhaps one for a cell phone. All of a sudden your simple flat file (one dimensional list) becomes a hierarchical one. Soon you are maintaining contacts by organization and by type of contact (social, work, and so forth), and what once was simple has become a complex data storage requirement.

Before XML there were few choices for storing hierarchical information. One could create a proprietary file format, use text–based file formats, or use a relational database. Of course, proprietary file formats are difficult to share without distributing the application needed to interpret them. Relational databases are powerful and well suited to maintaining mountains of hierarchical and relational data. However, in addition to being expensive and often difficult to distribute, they generally represent overkill for many small but complex data storage requirements.

Before XML, users were left with a text–based file format such as a comma–delimited file similar to the following. In this example field names are included in the first row to help identify the data.

```
"ID","First Name","Last Name","Email Address","Phone Number"
235,"Curtis","Young","curtis@softouch.com","2083222437"
275,"Mike","Young","mike@softouch.com","2083222437"
```

In this format, it is not clear how the list would appear to a user. The structure of the data is not separated by the type of data being presented. Other than a possible header row, the structure of the data is not preserved. The only way it could be accessed by another application is by position, that is, by row and column. With the text–based file format the data are presented to the user without real structure, making it difficult to reuse or transfer the data.

How can XML help? XML describes the data it contains by using elements and attributes. XML also allows the structure of the data, such as parent–child relationships, to be preserved as the data are exchanged. When XML is used to store and exchange data, applications can be more flexible because XML data can be accessed by position (as with flat files) or by name (as with databases). For example, the comma–delimited text file could appear as the following XML file. (Do not be concerned with the XML structure at this point. The second part of this chapter addresses related matters.)

```
<contacts>
<contact ID="235" FirstName="Curtis" LastName="Young">
    <email>curtis@softouch.com</email>
    <phone>2083222437</phone>
  </contact>
  <contact ID="275" FirstName="Mike" LastName="Young">
    <email>mike@softouch.com</email>
    <phone>2083222437</phone>
  </contact>
</contacts>
```

While the XML representation looks a bit more complex, it is easy to see the contact data structure. Notice that each contact is nested inside an element called contacts and that each individual contact has three attributes (ID, FirstName, and LastName). Each contact is further defined with two

63

nest elements: email and phone. Although nothing is visible to specify how many phone numbers are permitted, additional phone numbers could be added simply by adding additional phone elements.

XML brings several advantages to the table. Its syntax and formatting rules are simple, it is text–based, and it is an accepted standard. However, it is the ability of disparate systems and applications to read a common, universal file format that constitutes a solid step toward simplifying enterprise development. For the first time, a hierarchical data–based document with customized meta–information markup (customized element names) can be read by another program, even one written in a different programming language and running on a different operating system, without knowing what the file contains. Although that capability is useful in many situations, it is critical on the Web, where companies cannot control the systems of their information sources, business partners, and customers.

Managing Data Content

XML files describe data by using elements and attributes. They inherently say nothing about how the data should look. As a result, XML documents provide an excellent way to mark up data content in a platform– and language–neutral manner. Consequently, XML makes it possible to separate data from their presentation and business logic. This leads to numerous benefits, not the least of which is device independence.

By using XML to mark up data, you can publish data once and later transform the data to any other format you choose. With the use of technologies such as Extensible Style Sheets (XSS), XML documents can be transformed easily into a variety of other formats, such as HTML, WML, PDF, flat file, and electronic data interchange (EDI). The ability to work with XML on different platforms and change its structure to target different types of document formats makes it an excellent choice for use with content management applications.

While XML documents contain elements and attributes that are consistent with a few simple syntax rules, most XML documents require a valid structure. For example, in the sample shown above how many phone numbers are permitted? Are they optional? To enforce a structure, or valid "grammar," an XML document can be associated with another document that defines the rules for the elements it contains. One such document, the earliest one available, is called a Document Type Definition (DTD). This concept was powerful but had some weaknesses. While DTDs, like XML, are descendants of Standard Generalized Markup Language (SGML), the grandfather of markup language standards, they have a decidedly document, rather than data, focus. To enforce data structures more directly, another validation document called an XML schema was recommended.

Until XML schemas were available, relational databases had a huge advantage over XML in that they could store strongly typed data. The sample XML document shown earlier consists of text and numeric items, some stored as quoted strings. When you retrieve an item from a database, you usually know its data type.

XML schema documents define a common set of data type names, such as *string, integer,* and *date,* so that you can create a schema to accompany an XML document that defines the data types that each tag or attribute can hold. By reading the schema first, parsers can cast data values to the correct types as they read the data in associated documents. This is not possible with DTDs.

In addition, schema documents define boundaries for the order and arrangement of elements within conforming XML documents, the range or set of valid values for attributes, and the minimum and maximum occurrences of tags and attributes. Parsers that can read XML schema and compare the

content of XML documents with a schema are called validating parsers; an XML document that conforms to the rules of its schema is called a valid document.

Schemas are like a contract in that they define and enforce the format and content of the documents used on both ends of a transaction. Perhaps most important, schemas provide a way for machines to perform codification and validation processes declaratively and without human intervention, often without the need to write custom validation code.

Piercing the Firewall

XML documents and XML schemas, both of which are text–based documents, can be passed readily between heterogeneous systems. All that is required at both ends of the transaction is an XML parser, a tool that can interpret XML much as a browser interprets HTML. An XML document created on one system can be readily accessed by a program running on a different system.

However, since the document must be translated at both ends of the connection, passing XML across the Web has inherent inefficiencies. Earlier distributed applications were designed to transmit data efficiently because of bandwidth limitations. Typically, data were transmitted in binary formats, which required less translation. However, while binary formats are suitable for trusted connections, most companies are unwilling to let binary data flow in and out of a company over the HTTP port. In contrast, it is relatively safe to let text information flow back and forth, and this is exactly how XML behaves. XML documents are translated on the sender's side into a text representation, transmitted via HTTP, and then retranslated back into the original form on the server's side. Translating to and from text is safe but inefficient.

Additionally, XML documents can handle more than text, numbers, and dates: They also can represent data bound up in objects through XML object serialization. In combination with a schema, it is possible to save (serialize) an object and pass it to a remote machine where it can be deserialized (opened), effectively re–creating the object on the remote machine. Obviously, the programs involved need to understand how to use the objects, but object serialization significantly expands the data transmission capabilities of XML.

Even with its inefficiencies, the ability of XML to represent formatted data and serialized objects through firewalls using the HTTP port, a port that already is both exposed to the Web and protected by firewalls, routers, proxies, and virus scanners, is a significant benefit.

Calling Remote Components

With the enabling technologies, XML documents, XML schema, and object serialization to XML, it is possible to execute code on a remote machine. The technology to implement this is called SOAP. If two organizations agree on a schema, one machine can access another machine by sending an HTTP–encoded, XML–based message to a running service on the receiver's machine. The service will read the message and pass any enclosed message data to the appropriate component. Any response from the component is encoded in XML and returned to the calling machine. The message format needs to be simple, flexible, and easy to create.

To invoke the methods of remote components, one must know the location of the component and its available method names, parameter types, and return values. Two other emerging XML–based standards—Web Service Description Language (WSDL) and Universal Description, Discovery, and Integration (UDDI)—provide Web service discovery and description services. UDDI is a specification for building XML–document–based repositories for finding Web services. Companies store descriptions of their Web services in a repository, and so potential customers find the Web

services by searching the repository for specific business types, service types, or keywords. You can write a program to search the repository for the location of Web services that match your needs by searching the Web service descriptions available in the UDDI repository. UDDI repositories solve the location problem. After finding a Web service, a WSDL document describes the public interfaces (method names, parameter types, and return types) to the service.

XML and SOAP together form what is now called Web services. Web services in the .NET environment are discussed in more detail in Chapter 15, "Web Services with .NET."

Web Integration

More and more devices are supporting the use of XML. This allows Web developers to send XML data to many device types, such as personal digital assistants (PDAs), or to browsers such as Internet Explorer 5 and later versions.

Sending XML data directly to a device gives the end user control over displaying and manipulating the data. Instead of there being a need to make Web server requests each time data must be sorted or filtered, the XML data can be manipulated directly on the device without the need for a round trip to the server. The user also can make changes to the data and submit the XML document back to the server for processing.

Configuration Management

Many applications store configuration and application state data in a variety of files such as .ini files and the Windows registry. While these approaches work well, XML presents an excellent alternative for maintaining the data used by an application. Through the use of the .NET classes XmlDocument and XmlTextReader, these XML data can be read and integrated into applications. Using XML configuration files makes it easy to access the necessary data and allows applications to be updated and maintained more easily without recompiling code in many instances.

XML represents a revolution in data processing. The advantages that have been detailed here drove the rapid adoption of XML. The next topic to explore is what XML is: how it defines and manages data.

An XML Overview

As can be seen in the earlier part of this chapter, the features and benefits of XML are compelling. Every day more developers become convinced that XML will change the software industry dramatically. In fact, Microsoft, the world's largest software company, is betting the company on its .NET platform and, consequently, its fundamental underlying XML technology. However, as you begin to explore XML, chances are that you will find a bewildering array of acronyms that seem to be in a state of flux. In fact, the main obstacle to gaining a solid understanding of XML is its rapid evolution.

Moreover, until you understand how the pieces fit together, it will be extremely difficult to assimilate much of this material. A brief review of the World Wide Web Consortium XML Web site (www.w3c.org/XML) best illustrates what one is facing as one approaches this exciting technology. How does a user conquer it? As they say, "One bite at a time." See Figure 5.2 for an overview of using XML.
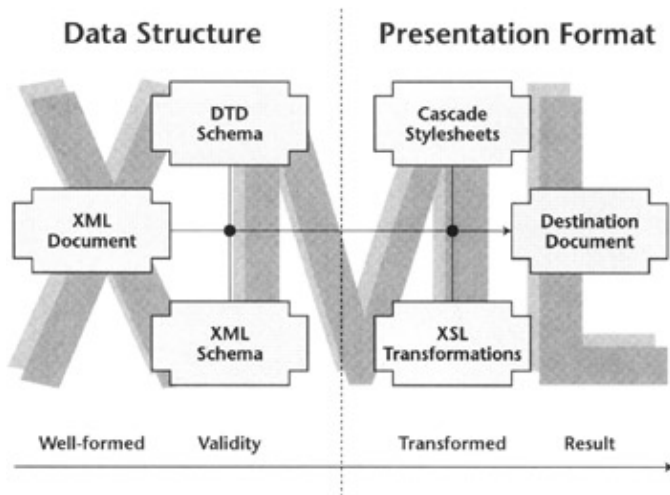
Figure 5.2: The process of using XML technologies to store, manage, transform, and present XML data.

The following sections introduce the reader to XML and its supporting technologies Each of the functions represented in Figure 5.2 is discussed in the sections that follow. You already have been exposed to the significance of this technology; the next sections will explain the pieces and show how each one fits into the big picture.

XML Basics

XML was derived from SGML. Both XML and SGML can create self–describing documents. Both use textual markup tags, or elements, to describe data so that other applications can correctly interrupt the data elements being presented. XML is a simplified version of SGML that is easier to learn and use and is more suitable for the Web.

XML defines the syntax for describing data. For example, the following is proper XML syntax:

```
<name>Carl Sagan</name>
```

Unlike other markup languages, XML is case–sensitive. The <name> element is different from a <Name> element. However, it generally is not a good idea to differentiate data elements by case alone. Also, XML does not guess about white space. It generally does not ignore white space as other languages do but instead preserves spaces as part of the data being defined. Additionally, in working with XML you must use special characters carefully. As with most languages, certain characters are used for syntax and, if used in the data they are defining, could confuse an XML parser.

An XML document is considered *well formed* if it contains exactly one root element, often called the document element, and all its child, or nested, elements are properly opened and closed within each other. This means that both the begin and end tags of a given element should exist within the body of the same parent element. The following is an example of a well–formed XML document. Note that it commences with an XML declaration (content between <? and ?> representing processing instructions to the XML parser) and then includes an attribute, named id, for the product element.

```
<?xml version="1.0"?>
<products>
  <product id="235">
    <description>Labtec Speakers</description>
    <vendor>Next Choice</vendor>
    <phone>2083755231</phone>
```

67

```
    </product>
</products>
```

The XML syntax just reviewed looks similar to the Hypertext Markup Language (HTML). Both use the same markup language syntax to define begin and end tags and attributes. HTML can be considered a loosely defined case of an XML language with predefined elements and behavior. These elements and their associated behaviors define how a given document will be rendered within a Web browser and used by the end user or application.

In a manner similar to the way HTML provides a methodology to create user interfaces, XML offers a methodology to describe and work with data. XML allows developers to create their own data vocabularies to describe their particular data elements and the related structures. If a developer is writing software for a training company, an element that contains information about contacts might be helpful in marketing the company's services.

As was discussed earlier, once a developer describes data in XML, the developer can easily exchange the data with other systems even if those systems are running on a different operating system. The developer also can receive data from another system as long as the data are defined in XML. Developers using XML no longer need to worry about platform, operating system, language, or data store differences when communicating with other systems. XML has become the lowest common denominator.

XML Namespaces

Because XML is about system interoperability and developers are free to define their own XML elements, the potential for developers to create identical element names for different data elements is very real. To prevent these potential conflicts, namespaces were introduced into the XML language.

XML namespaces provide a context for XML elements that allows developers to resolve elements to a particular definition. In the products example, on one system the price element might represent the sales price, but on another system it might represent the store's purchase price. The following example illustrates how namespaces can help resolve this kind of ambiguity:

```
<?xml version="1.0"?>
<products
    xmlns:purchase="http://vendor.com/prices"
    xmlns:sales="http://customer.com/prices">
  <product id="235">
    <description>Labtec Speakers</description>
    <vendor>Next Choice</vendor>
    <purchase:price>0.99</price>
    <sales:price>2.99</price>
  </product>
</products>
```

As you can see, XML syntax is not difficult to understand. Leveraging XML data and doing something useful with the data are much more challenging. To do this, one has to be able to process the XML file programmatically. This is the job of the XML processor, a software module that is capable of reading XML documents and providing access to their content and structure.

One of the main advantages of adopting XML is that any XML processor should supply the functionality needed to accomplish this goal. Developers do not have to write their own XML processors but can choose the best processor for their particular requirements and not worry about

compatibility issues.

A standard XML processor can interact programmatically with an XML document. If the XML document was created on a Windows–based system, it can be operated on by an XML processor on a mainframe system and in a Unix environment. While XML will not solve all software problems, it is an effective tool for exchanging data.

XML Core Technologies

Although XML documents are valuable in isolated situations, the real power of XML can be exposed through integration with its supporting technologies. Although you may not use all the technologies covered in the remainder of this chapter, you should have an appreciation of them as you design your overall XML strategy.

Validation Technologies

As you have seen, XML syntax is simple and flexible. However, the fact that an XML document is syntactically correct does not mean that it is "grammatically" correct. In other words, just because an XML document follows the rules of proper XML syntax (there is a single root element, nested elements open and close within the parent element, and so forth) that does not necessarily mean that the data within the XML document are represented properly. Just as there are rules to define the proper syntax for an XML document, there are means to define the proper format or structure for specific XML documents. When an XML document is syntactically correct, it is said to be *well formed.* When the data represented in an XML document adhere to a defined structure or format, the data are said to be *valid.* For example, the following code segment illustrates a well–formed but potentially invalid XML document.

```
<?xml version="1.0"?>
<products>
  <product id="235">
    <product id="235b">
      <vendor>Next Choice</vendor>
      <description>Labtech Speakers</description>
      <price>249.00</price>
      <price>395.00</price>
    </product>
  </product>
</products>
```

Even a cursory review of this XML document fragment indicates that it may have some problems. For example, there is a product element within another product element. While there is nothing inherently wrong with this XML structure (it is syntactically correct), it does not make much sense. There are multiple price elements within the inner product element. Which one is correct? Let us examine a couple of technologies for validating an XML document against a set of predefined rules.

A schema is a set of predefined rules that describe a given class of XML documents. A schema defines the elements that can appear within a specific XML document, along with the attributes that can be associated with a specific element. It also defines structural information about the XML document, such as what elements it may contain and whether those elements consist of data, other nested elements, or nothing (that is, the document is empty). The sequence of nested elements and the number of elements permitted also are defined by a schema. The two predominant schema types are DTDs and XML schemas. Both are specifications that outline how an XML document must be formatted and first require that the XML document be well formed.

The DTD language was created specifically to define validation rules for SGML documents. Since XML is a simplified subset of SGML, DTDs also can be used to define XML validation rules. An XML processor can use the DTD at runtime to validate a given XML file against the predefined XML schema.

The DTD syntax can be complex. DTDs use different syntactical elements, such as exclamation points, parentheses, asterisks, and angled brackets, to define the required elements of an XML document, optional elements, how many occurrences of a given element are allowed, and so forth. DTDs also describe the relationship between elements and show how attributes relate to specific elements.

Following is the DTD (product.dtd) for the example product.xml document.

```
<!ELEMENT products (product)*>
<!ELEMENT product (description, vendor, price)>
<!ATTLIST product id CDATA #REQUIRED>
<!ELEMENT description (#PCDATA)>
<!ELEMENT vendor (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

This document states that the *products* element can contain many *product* elements. Each product element must contain an id attribute and three nested elements, all of the type #PCData (parsed character data).

Documents that conform to this DTD would then add the following line. This statement tells the parser to validate the content of the XML document against the schema described in the DTD:

```
<!DOCTYPE products SYSTEM "product.dtd">
```

Although many XML processors support DTDs, DTDs do not do justice to the task of validating actual data content. Additionally, the syntax is unusual and inconsistent with XML syntax. While it is fortunate that some XML processors support the DTD syntax for describing schemas along with the XML syntax for reading documents, this is an inefficient process. If it were possible to describe schemas using XML, validation would be more efficient and easier for developers to work with. The W3C recently extended recommended status to XML schemas, a richer validation schema for XML documents.

An XML schema provides a superset of the capabilities found in a DTD. Both provide a method for specifying the structure of an XML element. Whereas both XML schemas and DTDs allow for element definitions, only XML schemas allow the developer to specify type information. Let us enhance the product example to demonstrate the advantages of schemas over DTDs:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Products">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Product" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="description" type="xsd:string"/>
                <xsd:element name="vendor" type="xsd:string"/>
                <xsd:element name="price" type="xsd:decimal"/>
            </xsd:sequence>
            <xsd:attribute name="id" type="xsd:string" use="required"/>
          </xsd:complexType>
```

```
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:schema>
```

Elements and attributes are defined in an XML schema by specifying <ElementType> and <AttributeType> elements, respectively. They provide the definition and type of the element or attribute. An instance of an element or an attribute is declared by using <element> or <attribute> tags. One can define how many occurrences of a given element are allowed through the minOccurs and maxOccurs attributes. The XML schema structure defines where the element is allowed to exist within the XML document (a <products> element contains <product> elements, and so on).

Processor (API) Technologies

A developer must be able to access XML data programmatically to do something useful. Software that is capable of reading XML documents and providing access to their content and structure is referred to as an XML processor. By accepting the industry standard Application Programming Interface (API), a developer can write code for a given API implementation that should run under any other compliant implementation of the same API without modifications.

The Document Object Model (DOM) is a defined standard for programmatically accessing the structure and data contained in an XML document. It is based on an in−memory tree representation of the XML document. When an XML file is loaded into the processor, it builds an in−memory tree that represents the document, as shown in Figure 5.3. The DOM also defines the programmatic interface (the attributes, methods, and required parameters) used to traverse an XML tree programmatically and manipulate the elements, values, and attributes.
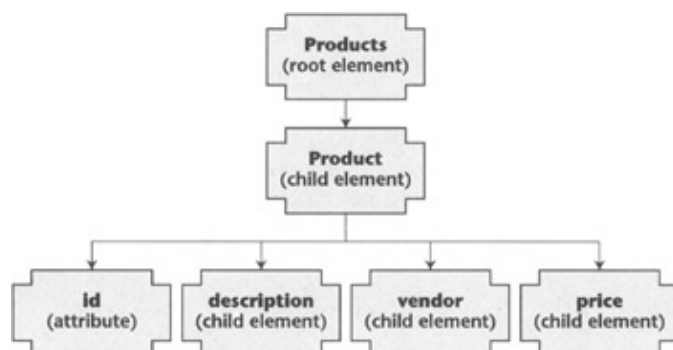
Figure 5.3: With the DOM API, an XML document is loaded in memory in a tree structure.
DOM provides an easy−to−use object model for interacting with the in−memory tree. The following Visual Basic .NET example illustrates how one could traverse all the child elements of the root document element in an XML document:

```
Dim doc as New XMLDocument()
Dim nodes as New XmlNodeList()
Dim node as New XmlNode()
doc.LoadXML("product.xml")
nodes = doc.ChildNodes
For Each node in nodes
     'process instructions for each node
Next
```

Transformation Technologies

Programmatically interacting with XML data through the DOM API can be tiresome, particularly as one attempts to extract specific data from a large XML document. A worse problem is attempting to

present the XML data in another format, such as HTML or another XML document bound to a different schema.

To simplify and standardize the process of performing these functions, a specification for XML transformations called the Extensible Stylesheet Language (XSL) was developed along with a simple query language called XSL Patterns.

Let us look at XSL Patterns, the query language, before examining XSL. A pattern references a set of nodes in an XML tree. The selection is relative to the current node to which the pattern is applied. The simplest pattern is an element name; it selects all the child elements of the current node with that element name. For example, the pattern product selects all the product child elements—description, vendor, price, and so forth—of the current product.

The pattern syntax allows you to identify the context in which a given element is within a document (for example, a price element that is nested within a product element). It also provides a filtering syntax to help identify specific nodes that match a given criterion (for example, where vendor=Next Choice). To find all the Next Choice products price elements within a products element, you would use the following pattern:

```
/products/product[vendor="Next Choice"]/price
```

When a pattern is applied to a given node, it simply returns a list of nodes that match it. You no longer need to traverse the entire XML tree programmatically to collect the specific data for which you are looking.

While XSL Patterns help identify certain nodes and the related data they contain, within a given XML document it is still up to the developer to do something with the data. For example, suppose you wanted to take your XML data and transform those data into HTML for browser rendering. XSL originated from this transformation requirement.

However, XSL goes far beyond that purpose. It is useful for defining transformations from one XML format to another XML format that is bound to a different schema. This makes interoperability much more feasible. If someone sends you an XML document that does not use the exact XML vocabulary, or schema, your system expects, you simply perform an XSL transformation and convert to a format consistent with your schema. With XML, developers no longer have to agree on a universal vocabulary for describing a certain type of data.

An XSL file contains a list of templates that define the transformation rules. Each template defines how you wish to transform a given node from the source document to a node (or some other type of data) in the output document. You use XSL Patterns within a template to define which parts of the document the template will transform. For example, to transform the following product XML file:

```xml
<?xml version="1.0"?>
<products>
  <product id="235">
    <description>Labtech Speakers</description>
    <vendor>Greasy and good.</vendor>
    <price>249.00</price>
  </product>
</products>
```

into the following HTML file:

```html
<html>
```

```
<body>
<h1>products</h1>
<ol>
  <li> Labtech Speakers, $249.00, Next Choice</li>
</ol>
</body>
</html>
```

you would use the following XSL file:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl=" http://www.w3.org/1999/XSL/Transform ">
  <xsl:template match="/">
  <html>
  <body>
  <h1>products</h1>
  <xsl:for-each select="products[vendor="Next Choice"]>
    <li><xsl:value-of select="description"/>, <xsl:value-of
select="price"/>,
      <xsl:value-of select="vendor"/></li>
    </xsl:for-each>
    </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

The first thing to notice about the XSL style sheet is that it is an XML document. XSL style sheets, like XML schemas, follow XML syntax. Next, note how XSL Patterns are used to match and select different XML elements. The text within the <xsl:template> tags defines the transformation rule for that set of elements.

It soon becomes apparent how powerful these transformation processes can be. Once you know the format for a destination document, you can make your XML data conform to that specific specification.

Other XML Technologies

This section describes other XML technologies that may be useful. Although not everyone may use these items, they are helpful for certain scenarios.

XLink

XML linking (XLink) defines the syntax between XML links. In short, XLink is used to define the relationship between resources. The HTML anchor <A> element defines relationships between HTML pages and allows them to be linked. Users can retrieve additional or related data on the subject of the current page. They will continue to follow links until they find the best match for what they are looking for. XLink provides the same type of service for XML links.

According to the XLink requirements, an XML link is the definition of an explicit relationship between resources. The specific identification methods that locate various types of data, such as Uniform Resource Indicators (URIs) and XPointers, are discussed next.

Here is an example of a simple XML link:

```
<product xml:link="simple" HREF="http:// company.com/product.asp">
</product>
```

73

XPointer

As was noted above, XLink depends on different mechanisms for identifying resources, such as a URI, that you wish to link to. XPointer specifies the methodology for addressing the internal structures of XML documents. In particular, it provides for a specific reference to elements, character strings, and other parts of XML documents.

An XPointer consists of a series of location terms, usually relative to the location specified by the prior location term. Each location term has a keyword (id, child, ancestor, and so on) and can have arguments, such as an instance number, element type, and attribute. For example, the following XPointer refers to the second child element whose type is product:

```
child(2,product)
```

XHTML

XHTML is a family of new XML–conforming HTML documents. XHTML documents work in conjunction with XML processors. Most HTML files rendered in browsers today are not well–formed XML documents. For example, it is common to have a beginning line break tag (<BR>) tag without a corresponding end tag </BR>. Hence, it becomes difficult, if not impossible, to interact with HTML by using standard XML tools. XHTML documents are well–formed XML documents, and so they are viewed, edited, and validated with standard XML processors. This also makes it easier for lightweight clients (such as palm–size personal computers) to deal with error conditions. Consequently, XHTML documents are more likely to interoperate within and among various environments.

The XHTML family is the next step in the evolution of the Internet. By migrating to XHTML today, content developers can enter the XML world, with all its attendant benefits, while remaining confident of their contents' backward and future compatibility.

There are many other XML–related technologies besides the ones listed above. As this book explores Web services in Chapter 15, "Web Services with .NET," and Chapter 12, "BizTalk Server," you will get a deeper understanding of the specific XML technologies the .NET Servers more fully exploit.

Summary

XML is the communication medium for all .NET technologies and servers and as such is the foundation of the .NET Framework. Just as HTML is used as a common language for communicating vast amounts of disparate data between humans, XML is the mechanism for accomplishing the same task between business applications. Like HTML, this communication occurs regardless of the environment the applications were developed in or the operating systems they are running on.

One of XML's features, which is unusual for a markup language, is its attention to data rather than document structure. Most markup languages address the broader document structure rather than attempting to define its content. Unfortunately, once data are marked up, the data become inaccessible within their presentation. With the aid of DTD files and particularly XML schemas, XML now persists the data structure by employing separate processes for presentation and data structure.

In addition to defining a common communication medium and separating data and presentation, XML provides the mechanism to communicate that data safely between organizations. Since XML is

maintained in a text format and uses the HTTP protocol, as does HTML, XML data are permitted access through firewalls. This dramatically extends the power and utilization of enterprise applications that require connections across disparate and non–trusted platforms.

Later chapters will explain how the .NET Enterprise Servers exploit XML and its characteristics.

Review Questions

1. Why should you consider using XML?
2. What transformation options exist within XML?
3. What is the purpose of XLink?
4. What is an XML namespace?
5. What is meant by transformation, and how can XML be helpful in accomplishing this?

Chapter 6: Building .NET Components

Highlights

The .NET platform can be used to develop, deploy, and support distributed applications. This platform is tightly integrated and flexible, allowing developers to build enterprise class applications that in addition to incorporating new tools and technologies can leverage existing services and applications. Of course, this functionality is made possible by component–based development. Before beginning to build components in the .NET Framework, it is necessary to examine the role of components as building blocks in distributed applications.

Before .NET, distributed applications were created with Component Object Model components. Those components generally were in the form of compiled dynamically linked libraries (DLLs) that relied on setup programs or savvy support personnel to assure that they were properly installed. The DLL would host the COM components in a set of files that held only compiled content; those files could be placed anywhere on the users' system as long as they were referenced properly from the registry.

COM provided the first standard for reusing code between applications. When a component's interface or Application Programming Interface (API) was exposed, other applications could create an instance of the component and then manipulate it. As long as the component followed the COM specification, any client using a COM–compliant language could access the component's functionality.

Since several clients could access many of these components, communication standards between the client and the component were required. This becomes particularly complex when clients communicate cross–process and even cross–machine. The COM specification defined the plumbing required for this communication, and although developers had many tools to assist them in implementing the plumbing, this process still represented additional overhead that distributed applications had to deal with. Creating COM components' communication plumbing for an application could be an arduous task.

The nature of component communication under COM prohibited direct interaction between components. There were always the "stub" and "proxy" pieces, among other plumbing elements, that communication would pass through. While this process "tricked" the components into thinking they were communicating in–process, thus making the development of individual components easier, it rendered it impossible to subclass a component effectively without having the source code. In other words, while a component could be deployed for reuse across applications in a COM–compliant environment, the functionality of the component could not be used as a starting point for enhancing or extending its functionality. Cross–language inheritance was not available.

.NET components, in contrast, are built on a common foundation: the .NET Framework's common language runtime, which is discussed in Chapter 4, ".NET Services for the Development Environment." All .NET components can communicate directly with one another, eliminating the need for the plumbing required in the COM environment. Additionally, .NET components can be enhanced or extended directly regardless of the language in which the original component was developed. The .NET Framework permits cross–language inheritance. Components developed in C# can be subclassed (enhanced or extended) by components written in Visual Basic .NET, for example.

This chapter examines how to create .NET components and use them in .NET applications.

Specifically, this chapter does the following:

- Discusses the nature of distributed applications and the role components play
- Examines the architecture of .NET components
- Examines the nature and makeup of assemblies
- Discusses calling COM components from .NET applications
- Discuss calling .NET components from COM components
- Shows how enterprise services are used in .NET components

Overview of Distributed Applications

The fundamental principle of distributed application development is the logical division of an application's three fundamental services: presentation, business, and data services. Each service layer is responsible for specific application functionality and uses .NET components and other resources to varying degrees in rendering that functionality.

A typical distributed application is physically deployed with a client, most often a thin browser client that communicates with a middle tier, or physical layer. The middle tier typically consists of .NET components that enforce business logic on an application server. This tier communicates with a database that provides persistent data services for the application. Let us examine the individual layers and assess their reliance on .NET components.

Presentation Services

The presentation service layer is responsible for communicating with the application's user. It collects required information from the user, responds to the user's actions, and presents information to the user, most often on the screen but also through other media. This layer can take the form of a rich client or a thin client.

Rich clients take full advantage of local system resources in providing their services. These services most often involve interaction with Win32 API, the built–in Windows infrastructure on which all Windows applications are built. In .NET applications this interaction is indirect (managed) through the .NET Framework's Windows Forms. Rich clients that access the Win32 API directly are unmanaged applications that do not have the full benefits of writing to the .NET platform. See Chapter 4, ".NET Services for the Development Environment."

Thin clients, typically using Web browsers, provide their services by consuming the resources of other layers, or segments, of the application. This is accomplished most frequently through ASP.NET, as described in Chapter 4, or through XML Web services, which are discussed in Chapter 15, "Web Services with .NET." While they do not take advantage of local system resources, applications that use a thin client architecture are often more portable and easier to maintain.

Rich clients make more extensive use of .NET components than do thin clients. Of course, thin clients exploit the .NET component functionality of other layers, most frequently the business services layer.

Business Services

The business services layer defines the distributed application. This layer defines the business rules and application logic of the application and enforces compliance with those rules. It is the hub of an

application, exposing its services to any number of clients seeking the functionally it provides. In fact, in many instances access to other needed application resources is granted only through this layer.

In addition to business rules and application logic, the business services layer often provides generic application functionality, such as transaction management and security. The distributed application developer typically defines application logic and business rules in .NET components, but application services may come from other sources. For example, in distributed applications this tier often is physically located on its own application server. From this server the application may provide generic application functionality through COM+ services such as transaction management and message queuing, among others, and use the .NET Framework to supply services such as security and memory management, many of which are accessed directly through developer–defined .NET components.

Most often this tier is the guardian of or the gatekeeper to the application's most precious resource: its data.

Data Services

Data services support both data access and data storage. This layer is responsible for the trafficking of data in and out of the database. These services often are physically deployed on both an application server hosting application logic components, or middle–tier services, and a database server storing application data. Through the use of technologies such as OLE DB and ADO.NET, much of the functionality housed on the application server is provided through .NET components.

Data services on the database server are generally in the form of stored procedures and therefore are not reliant on .NET components or even the .NET Framework. It should be noted, though, that SQL Server 2000 provides native support for XML, which is the markup language for data structures. See Chapter 15, "Web Services with .NET," for a more detailed discussion.

Components and Assemblies

Components are packaged in assemblies. Assemblies are the reusable, self–described, version–controllable building blocks of .NET applications. The simplest assembly is a single executable that contains all the information necessary for deployment and versioning. Assemblies enable the common language runtime to locate and securely execute code in components. In particular, the runtime uses assemblies for the following purposes:

- Establishing security and version identity for code
- Providing a scope for class and type resolution
- Locating classes to load

This chapter will examine the nature of components and then delve more deeply into the fundamental .NET building blocks of assemblies.

Component Architecture

The .NET Framework is a library of components, along with supporting classes and structures, designed to make component development easy and robust. Because .NET components must support a certain interface or descend from a certain class, they are defined more strictly than is the

case in COM.

.NET components can be constructed by using classes and structures. These classes and structures, collectively referred to as types, represent object definitions that can be created, or instantiated, by a client when the functionality defined by the type is desired. Visual Basic .NET developers can also create .NET components by using modules. For the purposes of our discussion here, the focus will be on class and structure types.

Classes have the advantage of being reference types. A reference type uses a variable or identifier to hold the memory address hosting an instantiated object and the data it represents. Passing this reference to, or memory address of, an object is more efficient than passing all the data it may contain.

Structures, in contrast, are value types and do not require the allocation of memory in a way that must be managed. Because they are value types, the identifier contains the structure's data rather than simply a reference to it. As with classes, structures may have properties and methods, raise and handle events, and implement interfaces.

Which one should you define? If the type (class or structure) you wish to design represents a large instance size, that is, several properties whose data values must be maintained in memory, or you wish to enhance or extend the functionality of the object type, you should use a class. Structures with large instance sizes can degrade performance as the instance data are passed from method to method. Additionally, since you cannot inherit from a structure, structures should be used only for types you do not need to enhance or extend.

Let us take a closer look at what a .NET component is. Microsoft .NET applications are built from components. All .NET objects, which are defined and supported by a .NET component, expose members such as properties, methods, and events. As the developer of .NET objects, you also are responsible for implementing the interface (that is, the properties, methods, and events) necessary for other programmers to use your component's services. Much of your development time will be spent designing objects and writing the code that defines the objects and components exposed and used by your applications.

Typically, simple .NET object–oriented programming involves creating a class; adding the properties, methods, and events required by the class; and including the class in various applications. Component–based development takes this basic concept to a higher level. Although the components you build in .NET are based on object–oriented programming principles, they go beyond the simple classes that a developer might use in multiple applications.

A component is a special type of executable built from a .NET project. After compilation, the component typically is referenced by applications that need the services provided by the component. In many .NET Web environments, components run on the Web server and provide data and other services (such as security, communications, and graphics) to the Web services operating on the server. In a Windows Forms application a .NET component performs the same role that it performs on a Web server, but on a reduced scale.

.NET components provide a programmable interface that is accessed by client applications. The component interface consists of a number of properties, methods, and events that are exposed by the classes contained within the component. In other words, a component is a compiled set of classes that support the services provided by the component. The classes expose their services through the properties, methods, and events that constitute the component's interface.

Simple .NET programming involves not much more than creating a class; adding the properties, methods, and events required by the class; and using the class in different client applications. A .NET component, however, is a precompiled class module with a .DLL (dynamically linked library) extension. At runtime a .NET component is invoked and loaded into memory to be used by client applications. These .NET components most commonly are built and tested as independent .NET projects and are not necessarily part of another project. After compilation into a .NET DLL, these components become plug-in service providers for other .NET applications.

Each .NET DLL component may contain multiple classes. This means that each DLL may expose a single class or a variety of classes. For instance, if you build a .NET DLL that supports remote data access, the DLL may need to expose separate classes for the database and for the variety of DataSets also produced by the DLL. Alternatively, you may have a single DLL that supports error handling in your applications. In this case, the DLL exposes a single ErrorHandler class that performs all the error handling required by client applications.

Assemblies

Components are packaged in assemblies. Assemblies are the reusable, self-described, version-controllable building blocks of .NET applications. They form the fundamental unit of deployment, version control, reuse, and security. An assembly is a collection of types, classes, or structures defined in .NET components and resources built to work together and form a logical unit of functionality. An assembly provides the common language runtime with the information it needs to be aware of type implementations. To the runtime, a type, whether it is a class or a structure, does not exist outside the context of an assembly. The simplest assembly is a single executable that contains all the information necessary for deployment and versioning.

Assembly Overview

Assemblies are a fundamental part of programming with the .NET Framework. An assembly performs the following functions:

- It contains code that the common language runtime executes.
- It is the unit at which security permissions are requested and granted.
- It forms a type boundary. Every type's identity includes the name of the assembly in which it resides.
- It identifies the types and resources exposed outside the assembly as well as the other assemblies on which it depends.
- It is the smallest unit that has version control. All types and resources in the same assembly are versioned as a unit.
- It is the deployment unit.
- It is the unit at which side-by-side execution is supported.

Assemblies can be static or dynamic. Static assemblies can include .NET Framework types (interfaces and classes) as well as resources for the assembly (bitmaps, JPEG files, resource files, and so forth). Static assemblies are stored on disk. Dynamic assemblies run directly from memory and are not saved to disk before execution.

There are several ways to create assemblies. You can use development tools, such as Visual Studio .NET, that you have used in the past to create DLL or EXE files. You can use the tools provided in the .NET Framework SDK to create assemblies with modules created in other development environments. You also can use common language runtime API to create dynamic assemblies.

Assembly Versioning

One of the primary goals of assemblies is versioning. Assemblies are designed to simplify application deployment and solve the versioning problems that can occur with component−based applications. Specifically, assemblies provide a means for developers to specify version rules between different software components and have those rules enforced at runtime. Because assemblies are the building blocks of applications, they are the logical point for specifying and enforcing version information. Each assembly has a specific version number as part of its identity.

Many users and most developers have experienced versioning and deployment problems with today's component−based systems. They have all experienced the frustration of installing a new application only to find that an existing application suddenly fails.

The .NET Framework solves many of these deployment problems through assemblies. Because they are self−describing components that are not dependent on external reference sources or registry entries, assemblies affect only the applications with which they are deployed. This was not true in Win32 or pre−.NET applications.

Before .NET, versioning control relied on backward compatibility, which was often difficult to guarantee. Once they were published, interface definitions were immutable. Each element of a component's interface, name, parameter data types, and return types must maintain backward compatibility with previous versions. Complicating matters, components typically were designed to allow only a single version to exist on a computer at any given time.

Also, there is no way to maintain consistency between components that were built together and those which existed at runtime. As components were independently enhanced, typically for use with additional client components, associations with the original client components often were lost.

These versioning problems combine to create DLL conflicts, or what has been referred to as DLL hell. One application can inadvertently break an existing application because an installed software element was not fully backward compatible with a previous version. Compounding these problems was the lack of troubleshooting support to help resolve these problems.

To solve versioning problems as well as the remaining problems that lead to DLL conflicts, the runtime uses assemblies to do the following:

- Enable developers to specify version rules between different software components
- Provide the infrastructure to enforce versioning rules
- Allow multiple versions of a software component to be run simultaneously (side−by−side execution)

The specific version of an assembly and the versions of dependent assemblies are recorded in the assembly's manifest. The default version policy for the runtime is that applications run only with the versions they were built and tested with unless they are overridden by explicit version policy in configuration files.

Assembly Contents

There is no concept of an assembly as a file. Instead, an assembly is a collection of files that work together to provide a unit of functionality and are stored in the same directory on the disk. In general, as seen in Figure 6.1, an assembly can consist of four elements:

- The assembly manifest, which contains assembly metadata
- Type metadata, or member information about classes, interfaces, and structures
- Microsoft Intermediate Language (MSIL) code that implements the types
- A set of resources

Anatomy of an Assembly

Primary Entities

Assembly: Primary unit of deployment.

Module: Separate files making up an assembly.

Types: Basic unit of encapsulating data with a set of behaviors.

Resource: Additional uncompiled resources used by the assembly.

Assembly
Manifest

Module
Metadata

Intermediate Language

Type	Type
• Methods	• Methods
• Properties	• Properties
• Fields	• Fields

Resource
Graphic.jpg
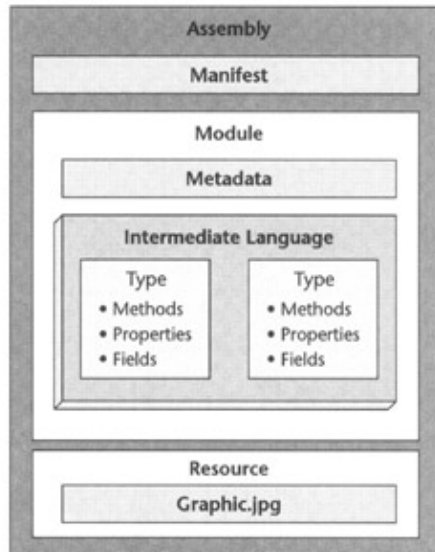
Figure 6.1: The structure and components of an assembly.

The information describing an assembly is called the manifest. In the most general case an assembly can consist of a number of executable files and resource files. In that case the manifest may be a separate file, or it may be included in one of the executable files. Assemblies enable the common language runtime to locate a code and execute it securely. In particular, the runtime uses assemblies for the following purposes:

- Establishing security and version identity for code
- Providing a scope for class and type resolution
- Locating classes to load

While only the assembly manifest is required, either types or resources are needed to give the assembly meaningful functionality.

Assembly Manifest

For the assembly to describe itself to the common language runtime, it must contain a set of metadata. This set of metadata is contained in the assembly's manifest and specifies the following:

- The assembly name.
- The assembly version, including major and minor version numbers.
- The assembly security information.
- The assembly file list, which lists each file making up the assembly.
- Type reference information, which maps the interfaces for all the types in the assembly.
- Reference assemblies, which list all other assemblies referenced by types in this assembly.

The manifest for an assembly can be stored in an EXE, a DLL, or a standalone file. In a single file assembly, the manifest is part of the file that constitutes the assembly.

The manifest of an assembly lists all of the assembly's files and resources. It also lists all the

classes and types defined within the assembly and specifies which resources or files they map to. The manifest also identifies any other assemblies that any elements of this assembly depend on.

Assembly File Structures

There are several ways to group these elements in an assembly. It is possible to group all the elements in a single physical file, as illustrated in Figure 6.2.

Single-file Assembly

MyAssembly.dll
• Manifest
• Type metadata
• MSIL code

Multi-file Assembly

Graphic.jpg

MyUtil.dll
• Type metadata
• MSIL code

MyAssembly.dll
• Manifest
• Type metadata
• MSIL code

Manifest
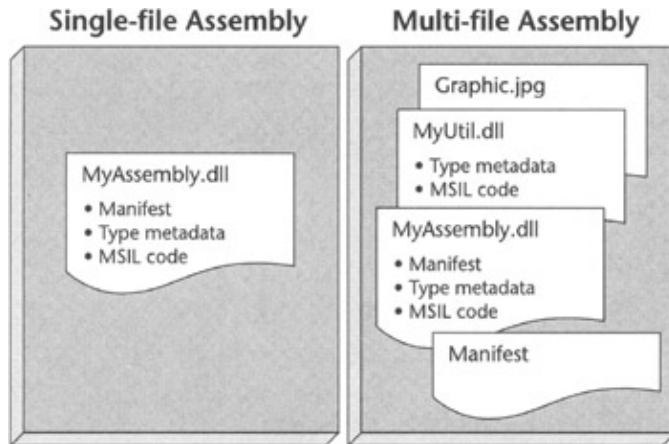
Figure 6.2: How assemblies are structured. Note that the assembly itself is not necessarily a single file. It can be a collection of files providing a unit of functionality.

Alternatively, the elements of an assembly can be contained in several files. These files can be modules of compiled code, resources such as BMP or JPEG files, or other files required by the application. You create a multifile assembly when you want to combine modules written in different languages or optimize the downloading of an application by putting infrequently used types in a module that is downloaded only when needed.

For example, Figure 6.2 illustrates a developer's use of a multifile assembly to separate some utility code and a resource file from the main file. In this case the .NET Framework optimizes the code download by using a file only when it is referenced.

The files that make up a multifile assembly are linked through the assembly manifest, and the common language runtime manages the files as a unit. The files are not physically linked through the file system. In this example all three files belong to an assembly, as described in the assembly manifest contained in MyAssembly.dll. To the file system they are simply three separate files. When the assembly was created, the assembly manifest was added to MyAssembly.dll, indicating its relationship with MyUtil.dll and Graphic.jpg.

In designing applications in the .NET development environment, partitioning the functionality of assemblies into one or more files will be among the optimization strategies a developer will use frequently. The developer also will make similar decisions concerning the partitioning of the application into one or more assemblies. Remember, just as components are elements of an assembly, assemblies are elements of an application.

Side–by–Side Execution

Side–by–side execution is the ability to run multiple versions of the same assembly simultaneously. The common language runtime allows multiple versions of the same assembly to run not only on the same computer but also within the same process. Support for side–by–side execution of different versions of the same assembly is an integral part of versioning and is built into the infrastructure of the common language runtime. Components that can run side by side do not have

to maintain strict backward compatibility, as is the case with the COM architecture. Because the assembly's version number is part of its identity, the runtime can store multiple versions of the same assembly and load the appropriate version at runtime.

Although the runtime provides the ability to create side−by−side applications, side−by−side execution is not automatic. There are precautions the developer needs to take when having components execute side by side. For example, external dependent resources that the component draws on may need to be maintained separately or identified by each executing component to assure that the proper resource is used. When one is running components side by side in the same process, even more precautions are necessary to assure that there are no conflicts or resource contention on process−wide resources such as database connections.

Working with Unmanaged COM Components

The discussion so far has dealt with creating new applications and components that work within the .NET Framework. However, for some organizations, existing COM components represent valuable resources for current and even future applications.

When a .NET application is running, the common language runtime is executing the MSIL code, not true machine language code. Any code executing under the common language runtime is called managed code. Unmanaged code, in contrast, is code executing outside the common language runtime. This would include COM components and any direct Win32 API calls.

Fortunately, switching from COM to Microsoft .NET does not involve a radical loss of existing components. In fact .NET supports two−way interoperability when working with COM components to avoid the loss of code base or productivity:

- .NET components can call COM components.
- COM components can call .NET components.

This two−way interoperability is the key to moving from COM to .NET. As you are learning the .NET environment, you can continue to use COM components. There are a number of situations in which this is useful:

- You will not know everything about .NET instantly.
- Some COM components may not be easily ported from Visual Basic 6 code to Visual Basic .NET.
- You can prudently migrate a large application over time rather than attempt a complete migration all at once.
- You may be using third−party COM components for which you do not have source code.

Next, this chapter looks at working with COM components from the standpoint of both using COM components in a .NET application and using .NET components in a COM application.

Calling COM Components from .NET Clients

Code executing within the common language runtime is managed code and has access to all the runtime's services, such as cross−language integration, security and versioning support, and garbage collection. Since all COM components are by definition unmanaged code, they cannot use any of these runtime services. The problem with mixing managed code and unmanaged code in the .NET environment is that managed components expect the components with which they interact to

depend on the runtime just as they do.

The way out of this dilemma is to use a proxy. In general terms, a proxy is software that accepts commands from a component, modifies them, and forwards them to another component. The type of proxy used in calling unmanaged code from managed code is known as a runtime–callable wrapper (RCW). These wrappers provide the glue between managed code and unmanaged code. RCWs straddle the boundary between managed code and unmanaged code (see Figure 6.3).

Calling COM from .NET Clients
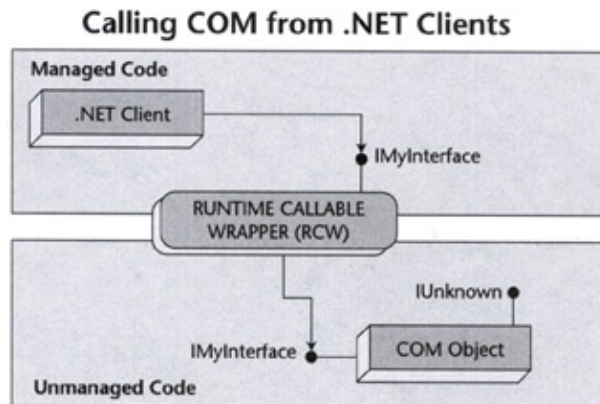
Figure 6.3: Calling a COM object from a .NET client.

The job of an RCW is to convert COM metadata to .NET metadata. A tool for performing this conversion is called a tlbimp (type library importer) and is part of the .NET Framework Software Developer Kit (SDK). Tlbimp reads the metadata from a COM type library and creates a matching common language runtime assembly for calling the COM component.

tlbmp is a command–line tool with a number of options for things such as signing the resulting assembly with a cryptographic key and resolving external references in the type library. The most important option is /out, which lets you specify a name for the resulting .NET assembly. For example, to convert a Visual Basic ActiveX DLL named support.dll to a matching .NET assembly with the name NETsupport.dll, you could use this command line:

```
tlbimp support.dll /out:NETsupport.dll
```

Although calling a COM component from managed code is simple, you should not consider this a permanent solution for most applications. Eventually you will want to convert your components to native .NET code. This will provide the full range of .NET capabilities and make the code faster by eliminating the wrapper layer.

When you start to move your development from Visual Basic to Visual Basic .NET, the ability to call COM components from .NET code becomes essential. Visual Basic .NET seemingly permits you to directly reference and call a COM component. Under the sheets, however, Visual Basic .NET is creating the wrapper class to facilitate its use in the .NET application. Treat this technique as an important part of the migration strategy.

Calling a .NET Component from a COM Component

Although COM clients can call code that is exposed in a public class by .NET servers, .NET code is not directly accessible to COM clients. To use .NET code from a COM client, you need to create a proxy known as a COM–callable wrapper (CCW).

As has been explained, managed code components not only depend on the common language runtime, they require the components with which they interact to depend on the runtime. Because

COM components do not operate within the common language runtime, they are unable to call managed code components directly.

As was noted in the previous section, the way out of this dilemma is to use a proxy. In this case, the type of proxy used for calling managed code from unmanaged code is known as a COM–callable wrapper. Figure 6.4 shows schematically how CCWs straddle the boundary between managed code and unmanaged code. Note the COM interfaces necessary for the communication to occur.
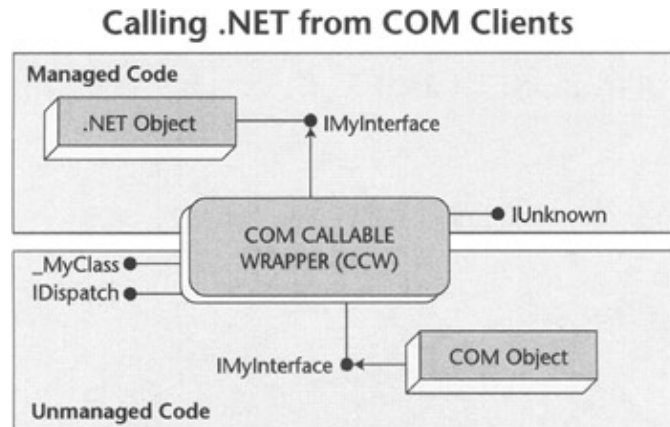
Figure 6.4: A COM client accessing a .NET component. The CCW proxy must expose the necessary COM interfaces for this call to be possible.

There are two prerequisites in creating a .NET class that will be used by a COM application. First, you must explicitly define an interface in your Visual Basic. NET code and have the class implement that interface. For example, the following code defines an interface named iFile and illustrates a class that implements that interface:

```
Public Interface iFile
    Property Length() As Integer
End Interface

Public Class TextFile
    Implements iFile
    ' implementation code
    ' for the Length property
End Class
```

Second, any class that is to be visible to COM clients must be declared public. The tools that create the CCW only define types based on public classes. The same rule applies to methods, properties, and events that will be used by COM clients.

After you have created a .NET assembly containing a class that will be called by a COM client, three steps must be taken to make the class available to COM.

First, you must create a type library for the assembly. A type library is the COM equivalent of the metadata contained in a .NET assembly. The .NET Framework SDK includes a tool called tlbexp (type library exporter) that can create a type library from an assembly. For example, using the /out option of tlbexp, you can extract the metadata from an assembly named NETServer.dll to a type library named NETServer.tlb with the following code:

```
tlbexp NETServer.dll /out:NETServer.tlb
```

Second, you should use the assembly registration tool (regasm) from the .NET Framework SDK both to create the type library and to register it in a single operation. This is the easiest tool to use

when one is doing both .NET and COM development on a single computer. To create and register a type library using regasm, use a command line such as the following:

```
regasm /tlb:NETServer.tlb NETServer.dll
```

Third, you must install the .NET assembly into the global assembly cache (GAC) so that it will be available as a shared assembly. To install an assembly into the GAC, use the gacutil tool, as in the following example:

```
gacutil /i NETServer.dll
```

The .NET component is now available to be called from COM clients.

Enterprise Services in .NET

The .NET common language runtime is Microsoft's next–generation component technology. While the runtime is a replacement for COM, it is not a replacement for COM+. COM+, which now is called .NET Enterprise services, is a runtime environment for objects that provides a set of utility services to facilitate the building of scalable distributed systems.

COM provides one way to write component–based applications. COM+ is not a new version of COM but a services infrastructure for components. Once they are constructed, components can be installed in COM+ applications (or what was earlier called Microsoft Transaction Server (MTS) packages) to consume these services. If a component does not need to use any services, it should not be placed in a COM+ application. Scalability and resource management are achieved by designing applications from the outset to take advantage of services such as transactions monitoring and connection and object pooling.

The COM+ services infrastructure can accessed from managed and unmanaged code. Services in unmanaged code are known as COM+ services. In .NET, these services are referred to as Enterprise services. This services infrastructure has little to do with COM or even with components: COM+ services now can be applied to COM components, to .NET components, and even to other entities that are not considered components, such as Active Server Pages (ASP).

All COM+ services that are available today are available to .NET and COM objects. Some of these services include transaction monitoring, object pooling, Just In Time (JIT), synchronization, and role–based security. For a complete listing of services on Microsoft Windows 2000, see "Services Provided by COM+" in the platform SDK. Microsoft Windows XP includes a new version of COM+ that has additional services that also can be used with .NET components.

Associating Components with Enterprise Services

All COM+ services are implemented in terms of contexts. A context is a space within a process that provides runtime services to the object or objects referenced there. Classes that use COM+ runtime services are called configured classes and are marked with one or more declarative attributes. These attributes define the COM+ runtime services the components are designed to use. COM+ stores and manages a configured class's declarative attribute values in a metadata repository called the catalogue.

When a configured class is instantiated, the object creation process has to ensure that the new object ends up in a content that is consistent with its requirements for COM+ services. At the time

when an object is instantiated, the catalogue is interrogated to determine whether it is defined as a configured class. If the new object has COM+ service requirements and they can be satisfied in the creator's context, that is where the object will reside. If the creator's context does not meet the new object's requirements, a new context for that object will be created to provide the COM+ services it requires. When nonconfigured classes are instantiated, the objects are always placed in the creator's context.

When an object in one context calls an object in another context, the call is intercepted by a proxy. The proxy gives the COM+ runtime an opportunity to execute the necessary service code to satisfy the object's COM+ requirements, such as transaction management and object pooling.

Memory Management with COM+ Resources

Since one of the primary reasons for using COM+ services is to manage the use of scarce resources effectively, it is imperative that those resources be released as soon as they are no longer required. Of course, one of the major features of the common language runtime is its reliance on garbage collection to reclaim memory. While this helps in better managing memory, it makes managing other types of resources more difficult.

The problem is that some resources, such as database connections, are very precious and should not be floating around in memory waiting for the garbage collector to reclaim them. .NET has provided a way to reclaim these resources immediately. Each .NET configured class implements an interface defining the Dispose() method. This method calls the object's finalizer method immediately rather than waiting for the garbage collector to do so.

Summary

All COM+ services are available to managed code. The .NET Framework now provides equal access to all COM+ services in a consistent and logical manner. Additionally, other elements of the .NET Framework, such as ADO.NET, messaging, and ASP.NET, integrate deeply when .NET components engage COM+ services, providing a consistent architecture and programming model.

Review Questions

1. What is the difference between a rich client and a thin client?
2. What roles do .NET components play in the presentation services tier?
3. With respect to .NET components, what is the difference between a class and a structure?
4. What role do assemblies play in .NET application development?
5. What is a manifest, and what is included in it?
6. What is the significance of side–by–side execution of .NET components?
7. What are Enterprise services in .NET?

Part II: .NET Enterprise Servers

Chapter List

Chapter 7: Exchange Server

Overview

Over the years Exchange Server has increased its market share. Many organizations have incorporated their email infrastructures into this Microsoft product. As the product has continued to grow in use, it has added several features that have enhanced its functionality. Exchange Server is still known primarily as a messaging server, but its use has expanded greatly into the development arena. Many of the new features of Exchange Server are centered on its data accessibility and collaboration features. The Exchange Server database has become more accessible to developers, and this is making it easier to develop .NET applications that interact with Exchange Server data.

A developer now can access the Exchange Server storage system easily to make mailbox and public folder data accessible through other applications and interfaces. This feature makes it possible to integrate Exchange Server data with data stored in other locations to develop an application that provides an interface to everything the user needs. The services that Exchange Server includes for .NET facilitate the sharing of data between .NET Enterprise Servers. BizTalk Server plays a particularly important role with Exchange Server, as it can be used to transfer data in and out of Exchange Server through XML.

Why is it beneficial to use Exchange Server instead of another repository, such as Structured Query Language (SQL) Server, as the repository for application data? Exchange Server provides features that are not readily accessible by most relational database management systems. Many developers have found it necessary to create workflow applications that automate the delivery of messaging from one location to another. For example, a developer may want to create a purchase order tracking system. In most organizations a purchase order must be approved by several people. Exchange Server makes it possible to automate the delivery of the purchase order request to a manager after the initial approval has been made. The messaging features available with Exchange Server make the development of a message–routing application such as this much easier. With the enhancements to Exchange Server 2000, these workflow applications are easier to create than they were in previous versions of Exchange Server. However, the real benefit is the option to reuse or transfer the data to another location. The new interfaces supplied with Exchange Server 2000 make data readily available to the other .NET Enterprise Servers. This allows for the integration of applications and data at levels that previously were unavailable.

This chapter discusses Exchange Server 2000 from the standpoint of .NET solutions. The first section identifies the different features associated with the various editions of Exchange Server. You want to make sure you have chosen the correct version of Exchange Server to take advantage of the features that are essential to your organization. The chapter then describes the configuration modes of Exchange Server. This is a key section because Exchange Server does not have to be a messsaging server. In most cases it is used in this way, but it also can be used as an application server to store data for a public folder–based application. The chapter then identifies the core features of Exchange Server that are enhanced by the services provided by .NET. Specifically, this section addresses the data retrieval options with ADO.NET and XML. These interfaces are made possible in Exchange Server 2000 by the Web storage system. The chapter ends with a section on the interaction of Exchange Server with the other .NET Enterprise Servers. This section presents a few examples of how Exchange Server can be used as a solution with the other products in the .NET Enterprise Suite.

Editions of Exchange Server

As with most Microsoft products, with Exchange Server one can choose between different editions. One can select either the Standard Edition or the Enterprise Edition of Exchange Server. This section identifies the differences between these two versions and can help you decide whether the Enterprise Edition is worth the additional cost. Both editions provide the basic messaging functions and supply most of what a user needs from a messaging system. Most of the features that differentiate the two editions do not affect application development. They do, however, affect the storage of the Exchange Server data, which could affect the stability and availability of an application.

Standard Edition

The Standard Edition of Exchange Server is significantly less expensive than the Enterprise Edition. It was designed to be installed on the .NET Server product. The core features needed for storing and retrieving messages are included with the Standard Edition:

- Messaging.
- Public folders.
- Integration with Active Directory.
- Outlook Web Access.
- Internet Protocol support, including Lightweight Directory Access Protocol (LDAP), Post Office Protocol version 3 (POP3), Internet Mail Access Protocol version 4 (IMAP4), HyperText Transport Protocol (HTTP), Simple Mail Transport Protocol (SMTP), and Network News Transport Protocol (NNTP).

Enterprise Edition

The Enterprise Edition is more expensive than the Standard Edition; it is also more functional. Most of the additional features provided with the Enterprise Edition increase the scalability and stability of Exchange Server. The Enterprise Edition includes all the features supplied with the Standard Edition and also has the following features. The Enterprise Edition was developed to be installed on .NET Advanced Server or Data Center Server.

- **Clustering services.** Through the cluster services of .NET Server and Exchange Server it is possible to create a fault–tolerant solution that provides protection if one of your servers fails. The responsibility of that server is transferred automatically to the other server. This is a premium service for applications that are mission–critical. Clustering is described in more detail in Chapter 18, "Clustering .NET Servers."
- **Multiple storage groups.** Exchange 2000 Enterprise Edition makes it possible to locate mailboxes and public folders in separate databases. This feature allows you to back up these databases independently and take a single database offline without taking down all the users. This feature eliminates one of the primary weaknesses of previous versions of Exchange and the current Standard Edition, which are limited to a single database for the storage of all mailboxes on a single server. Every mailbox has to be maintained as a group, resulting in longer backup times and lengthy restore operations. This is an exciting feature for application development. For example, if you created an application to track help desk requests and chose to use Exchange Server public folders as the database storage mechanism, the database used by the application can be separated from the other public folders and messages data in Exchange Server. This allows your application to be maintained independently of the other data on your server. If another database hangs, fails, or becomes corrupt, the application is not affected.

- **Multiple administrative groups.** Through administrative groups you can isolate a group of servers to be administered through the same set of permissions. If you have a large number of Exchange Servers, you can group them by administrative needs, not just messaging requirements.

Configuration Modes

With Exchange Server it is possible to strategically configure a server to play a dedicated role. In many organizations a single server handles all the responsibilities related to Exchange Server. However, as an organization scales, it may make sense to isolate individual servers to play a specific role for Exchange Server organizational requirements. As applications that store data in Exchange Server databases are developed, it may make sense to isolate the application from the Exchange Servers that are responsible for messaging. This is done frequently in voice mail and conferencing applications that use Exchange Server as their repository. The following list describes each of the different roles an Exchange Server can play, along with a short description of the required configuration:

- **Messaging server.** The default after installation is for the Exchange Server to support mailboxes. This is the mode that most people are comfortable with. Most of the Exchange Servers in an organization will be installed and configured for this purpose. Most users have purchased Exchange Server to handle email, and regardless of the application development features described in this book, messaging is what Exchange Server does best. As the role of Exchange Server grows within an organization, it is important to remember that its primary function still is to send and receive email messages.
- **Public folder server.** An Exchange Server can be configured to support only public folders. A public folder is a central repository to store messaging or application–related items. Public folders can be used as a storage mechanism for several different types of items, such as contact information, calendar information, workflow applications, informational forms, intranet site content, and client/server applications. This is the purpose of much of the application development with Exchange Server. You can use the public folder system as a repository for your applications. The data then can be retrieved through one of several different interfaces, including XML, ADO.NET, and Messaging Application Programming Interface (through an email client).
- **Front–end server.** A front–end server is a solution for Web messaging requirements. You can install a front–end server outside the firewall, in this case Internet Security and Acceleration (ISA) server, and allow users from the Internet to connect to the front–end server. The front–end server then passes the request to the Exchange Server that sits behind the firewall that stores the mail. A front–end server does not store any mailboxes. Its sole purpose is to support Internet connectivity. The configuration of a front–end server from Exchange Server is fairly straightforward. To complete the configuration you also need to configure the firewall to allow the flow of communication between the front–end server and the Exchange Server that is behind the firewall. The details of the firewall configuration are described later in this chapter in the section *Integration with .NET Enterprise Servers*. You can enable a front–end server by performing the following steps:

    1. Open the System Manager from the Microsoft Exchange program group, as shown in Figure 7.1.
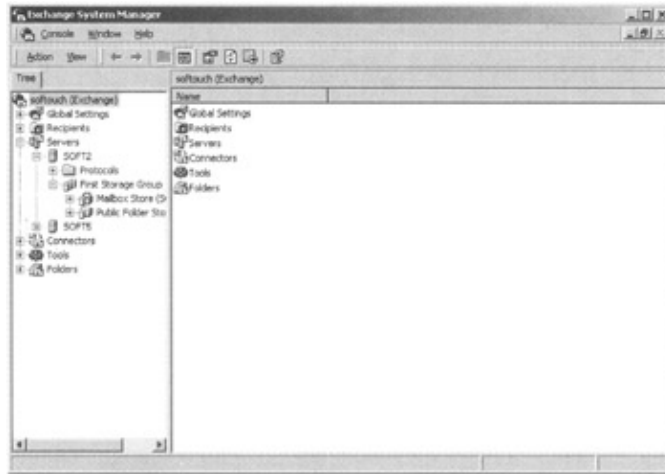
Figure 7.1: The System Manager is the primary configuration tool for Exchange Server 2000.

2. Click to expand servers.
3. Right–click the server you want to be a front–end server and select Properties.
4. From the Server Properties dialogue, click the checkbox labeled "This is a front–end server," as shown in Figure 7.2.
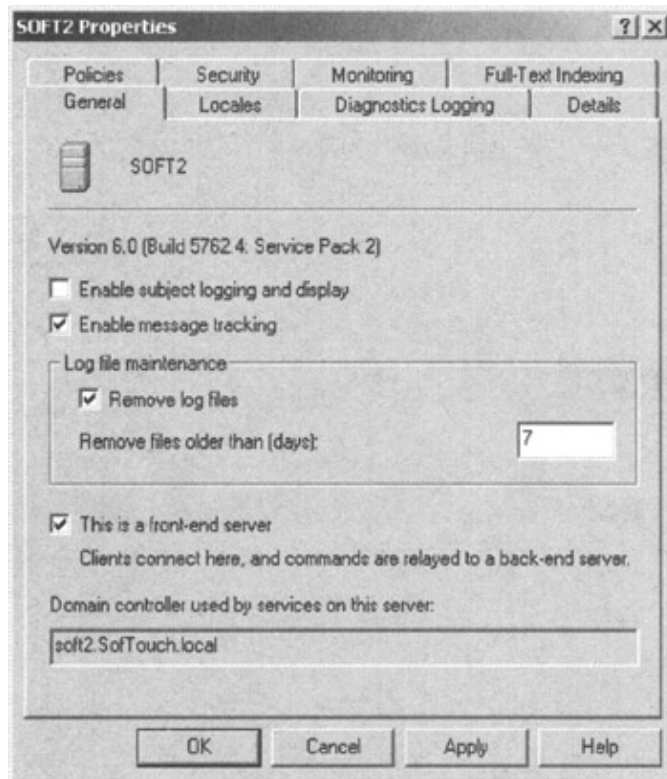
Figure 7.2: Front–end servers isolate Internet connectivity from mailbox servers. The front–end server cannot store mailboxes; it only accepts requests based on Internet protocols and interacts with mailbox servers to fulfill a request.

- **Conferencing server.** The Conference Server is technically not a configuration option within Exchange Server. Conference Server is a separate product that ships with Exchange Server. It can be installed on the same machine as Exchange Server or as a standalone. If it is installed on a separate machine, it is still dependent on Exchange Server and will connect to Exchange Server for many of its features. Conference Server can be used to support videoconferencing, audioconferencing, shared applications such as Net meeting, and chat services.

.NET Services

This section describes the relationship between the new .NET services and Exchange Server. It introduces the interfaces that can be used to retrieve and store data. Exchange Server is a little different from the other servers described in this book because some of its new features enable applications that have been available for some time in other database products. The .NET services that are provided or enhanced by Exchange Server 2000 are outlined in the following list:

- With previous versions of Exchange Server it was not possible to isolate a database for the use of a single application without dedicating a complete Exchange Server to the application. With storage groups, application development is more feasible with Exchange Server because one can protect one's application from the other applications deployed on that server.
- Data are more easily accessible from the Web. Through XML, WebDAV, and XMLHTTP it is possible to access the data stored in Exchange Server via the Internet. This allows the applications to be exposed in a way that previously was very difficult to do.
- The ExOLEDB provider (OLE DB provider for Exchange Server) makes Exchange Server data accessible from ADO.NET and OLE DB. While these technologies have been around for some time in most database management systems, this type of data access is new to Exchange Server 2000.

The following sections provide an in–depth description of these features of Exchange Server 2000. The first part of each section describes the role of storage groups in Exchange Server. This is not a .NET service or feature, but it is the key to application development. If you want to take advantage of the new .NET services for Exchange Server, you must understand the role of storage groups in application stability.

These sections then describe the Web storage system. In the section on the Web storage system, Web interfaces and database interfaces are described in depth. This includes a description of the role of XML in Exchange Server 2000 and the options for accessing data through ADO.NET.

Finally, these sections describe the Exchange Installable File System (ExIFS) and show how it is used for application development. ExIFS exposes data through the Windows .NET file system. This makes data from Exchange Server available through Win32 functions and through a normal navigation of a file system via an interface such as Windows Explorer.

Storage Groups

Predecessors to Exchange Server 2000 required that all mailboxes that resided on a single server be stored in the same database. Additionally, all public folders on a single server were stored in the same database. Consequently, if the database had a problem, all the users were affected and any downtime to the database meant downtime to all the users on the server. Furthermore, the public folder database was stored in the pub.edb file and the messaging database was stored in the priv.edb file, but both were managed by the information store service. If either one of these databases became corrupt, it was impossible to start the service, and therefore access to both files was prevented. This was a severe downside to application development with Exchange Server. A user could not consider deploying a critical application to Exchange Server because of the risk involved with the application stability. If you want to deploy more applications to Exchange Server, you should consider using storage groups to isolate one application from another.

With Exchange 2000 you can split your mailbox and public folder resources across multiple message databases. This allows you to perform backup and restore operations for subsets of messaging resources independently of each other. For instance, you can place the mailboxes of important management staff in a separate storage group and perform backups more frequently than you do for the mailboxes of ordinary users. If you need to restore a storage group, other storage groups can remain online. Splitting mailbox and public folder resources across multiple storage groups increases reliability, results in fewer lost work hours in cases of system failure, and reduces the time required to restore a system. This is beneficial to application developers because it now is possible to store application data in a public folder separate from the other public folders on the server. Therefore, if another application or folder fails, it will not affect your application. You should keep in mind the following suggestions for storage groups:

- Storage groups hold databases, which are referred to as *stores.*
- A single database can be either a mailbox store or a public folder store. A single database cannot store mailboxes and public folders.
- Each server can have up to four storage groups.
- Each storage group can have five databases.

It is possible to add additional databases to a single storage group. If you want to add a database to the default storage group, perform the following steps:

1. Open System Manager from the Microsoft Exchange program group.
2. Click to expand servers.
3. Click to expand the server to which you want to add the database.
4. Right–click the storage group to which you want to add the database, select New, and select the type of store you want to create, as shown in Figure 7.3.
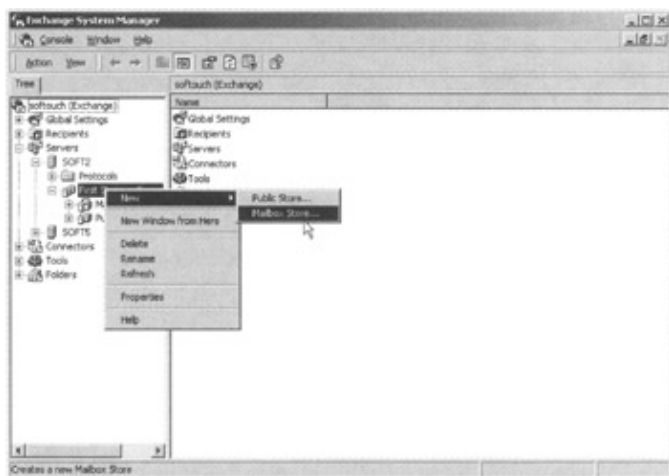
Figure 7.3: It is possible to add multiple mailbox stores to a single storage group. This makes it possible to store mailboxes in different databases to maintain databases efficiently.

5. Give the database a name and choose the file location for it.

Storage groups are beneficial for isolating mail and public folders from other databases on a system. For example, consider an employee time tracking system you are integrating with Outlook. You have created several forms that integrate into Outlook that your employees use to enter their daily time information. You initially store the information in a public folder. The data then are transferred from the public folder to an SQL Server database that stores your accounting and human resources information. You use the Data Transformation Services (DTS) of SQL Server to connect to Exchange Server via the ExOLEDB provider.

Your organization also has several other public folders that departments and various employees use casually. Some departments use these folders to share contacts, while others want to integrate calender information into them.

In this example you would be best served by storing your time−tracking public folder in a public folder store separate from other public folders to isolate your application from the other folders. Additionally, if one of the folders gets corrupted, it will not affect your application. By using multiple stores (databases) you can increase the availability and stability of your applications.

Web Storage System

The Microsoft Web storage system is an innovative approach to integrating the Information Store service of Microsoft Exchange 2000 Server with Web−based technologies. Through this integration you can take advantage of the .NET services with Exchange Server 2000. The Web storage system allows Exchange Server data to be accessed with interfaces such as HTTP, XML, Web Distributed Authoring and Versioning (WebDAV), Object Linking and Embedding Database (OLE DB), and ActiveX Data Objects (ADO.NET). This book focuses primarily on the accessibility of Exchange Server data from XML and ADO.NET. The Web storage system extends the accessibility of the information store to provide the following advantages for Exchange users:

- Platform−independent messaging
- Workflow application development
- Outlook Web Access (OWA)
- Data management and processing applications
- Integration with other Web−based applications

Because Exchange 2000 Server is based on relational databases that have a flexible database schema, the Web storage system can accommodate all possible data types. This allows you to store any type of data you want, such as messages, documents, contacts, appointments, audio, video, HTML files, and Active Server Pages (ASPs). Because this database provides storage for many types of data, you can use Exchange Server as a repository for all types of applications, not just those which are related to messaging functions. Because Exchange Server has been a messaging server for so long, people have a tendency to use it only as a messaging server. The Software Development Kit (SDK) for Exchange 2000 Server contains a set of development tools that enable you to design powerful Web−based business solutions quickly regardless of the data you need to store for your application.

The following sections break down the main development components associated with the Web storage system. The first topic addressed is XML integration with Exchange Server. Then the database interfaces are described. The database interfaces introduce OLE DB and then ADO.NET access to Exchange Server.

XML and Exchange 2000

The Web storage system relies on Windows 2000 Server and Internet Information Server (IIS) 5.0 to support Web browsers and WebDAV applications by means of HTTP, WebDAV, and XML. It is not necessary to configure anything extra on the server to provide users with access to mailbox and public folder resources.

When one is accessing data from an HTTP−based interface (XML is an example), the data have to be rendered to create the appropriate view for the browser. XML also is used to help identify the specific properties that are requested. This book includes a detailed description of the access of

data through XML. The following section explains the role of XML in Exchange Server.

XML Querying

XML simplifies the querying and modifying of item properties and other information contained in the Web storage system. The item is the object with which you want to interact. Examples of items in a typical Exchange Server include a mailbox, a contact, and a task. Properties are the descriptive information about the item. For instance, a mailbox has a display name, which is the name that appears in the global address list. Properties are based on various XML namespaces that group similar properties and ensure their uniqueness. When you access a property, you must specify the desired namespace, followed by the actual property name.

For instance, the DAV:namespace has a property called displayname that makes it possible to retrieve the display name of an item. You may, for example, want to see the display name of a mailbox. Another important DAV property is href, which holds the complete Uniform Resource Locator (URL) for an item. To access properties, programmers need to specify the full property reference, which is commonly known as the schema property name, such as DAV:displayname or DAV:href. By using these XML namespaces, you can retrieve the data you want from Exchange Server and display the data to a user or store the result in another database. It is common for developers to intero−gate Exchange Server to get pieces of relevant data out of the database without having to export the entire database. Some common information you may want to retrieve includes the following items:

- **Calendar information.**
- **Contacts.**
- **Postings to a public folder.** Public folders are a common repository for applications. Some common applications that use Exchange Server as a repository are help desk applications, purchase order tracking systems, and other applications that normally would be routed from one user to another to be analyzed or approved.

Retrieval of Data with WebDAV and XMLHTTP

You can retrieve XML data from Exchange 2000 in several ways, but you probably will use the WebDAV protocol most often. WebDAV is an extension to the HTTP protocol that specifies how to perform file processing, making it easy to read and write to the Web. WebDAV was first made available in HTTP Version 1.1. Most browsers in use today employ or support HTTP Version 1.1. By using WebDAV commands, you can lock a resource and get or change a property. Because it piggybacks on HTTP, WebDAV also can work through firewalls and proxy servers. WebDAV and XML generally run over port 80, the default port for HTTP. If the firewall allows port 80, it generally will allow XML and WebDAV requests. Recently this has become a security concern for many network administrators. Port 80 is now used for much more functionality than it was just a few years ago. Firewalls are now including functionality to scan HTTP data to determine which types of HTTP requests to allow. In the ISA server, Microsoft's Enterprise firewall solution, this is accomplished through XML Web filters. For more information on XML Web filters, refer to Chapter 14, "Internet Security and Acceleration Server."

One of the best examples of a Web application using WebDAV is the Microsoft Internet Explorer 5.0 or the later version of OWA that ships with Exchange 2000. OWA leverages WebDAV and the XMLHTTP object extensively. When you read email or write updates to a mailbox, such as a new message or contact, you are using WebDAV. Because most of OWA's source code ships with Exchange Server 2000, you can bring up those files and browse for code examples (using WebDAV, XML, and Extensible Stylesheet Language [XSL] from OWA) to use in your own Web applications. WebDAV is based on XML data encoding, meaning that the commands you send to

the server and the responses you get back are XML–based. You can find the source code files for OWA on the Exchange Server 2000 CD–ROM.

For example, you may want to extend the functionality of your OWA to include a new phone message form. You want a basic form to take messages for employees who are not currently available. You would like this form to be available from OWA just like a new contact or a new mail message. To complete this task you can open the source code files for OWA and point them to a new phone message form you have created. You also can extend OWA to include company–related items such as logos, presence information, and location information. You are not limited in your OWA presence to the interface that is provided with the product.

If you are wondering how to use the WebDAV protocol in your Web applications, IE5 ships with a component called XMLHTTP that helps you do that job. You still have to send correctly formatted WebDAV requests to get the results you want, but XMLHTTP simplifies the process. However, you should use the component on the client side only because it was built for that environment. The purpose of XMLHTTP is to render data for a client to view. If you are writing code on Exchange Server, you should write to OLE DB or ADO.NET. More information on XMLHTTP is available in IE5's product documentation. Additionally, XMLHTTP can be used to access data from a SharePoint Portal store. There is more information on the access of the Share–Point Portal store in Chapter 13, "SharePoint Portal Server."

The XMLHTTP object exposes several properties and methods that are used to interact with a server. Through XMLHTTP you can perform the following types of functions with your Exchange Server:

- Send requests
- Process the results of a query
- Connect to the HTTP server
- Check the status of a pending request
- Check for errors associated with a request from the server

Creation of Web–Based Forms

The Web storage system can handle electronic forms similar to the forms available in Outlook. Instead of Outlook forms, HTML– or ASP–based forms are launched when the user opens an item in a Web browser. WMTEMPLATES.DLL defines the default templates used by OWA to render standard HTML forms for email messages, contacts, and calendar objects. You can extend OWA by implementing a custom user interface that is based on Web storage system forms. You also can extend the functionality of OWA by incorporating your own forms into the Web storage system.

Web storage system forms must be registered in the Web storage system forms registry. Some common forms you may want to implement include the following items:

- Phone message
- Help desk request
- Supervisor review

This section identifies options and details for creating forms. The following items identify the things you should consider when implementing Web storage system forms:

- **You can employ Web storage system forms for Exchange 2000 Server SDK.** It is a good idea to install the Web storage system forms SDK on your Exchange 2000 server and the FrontPage Extensions for Web storage system forms on your workstation if you are

planning to design Web storage system forms. The server component integrates a custom forms engine (EXWFORM.DLL) into Exchange 2000 Server, which is used to render HTML forms with information from Web storage system folders. By using FrontPage 2000 with extensions for Web storage system forms, you can build Web–based applications directly in the Web storage system. This has several advantages. For instance, programming know–how is not required, custom properties can be defined conveniently, and HTML forms can be registered in the Web storage system automatically. Knowledge about Web storage system internals is not required. By using FrontPage, you can open Web pages directly in the Web storage system to edit them. This is a preferred method of customization and forms development for those who are not familiar with programming through Visual Studio .NET. The functionality is similar to the FrontPage Extensions Web on a standard file system or Web server. Detailed information about designing HTML–based Web storage system forms is available in the Web storage system forms SDK documentation.

- **You can use Active Server Pages to develop forms.** The Web storage system is capable of launching ASP.NET–based forms directly without the need for a custom forms engine. EXWFORM.DLL is not required to fill the Web page because ASPs can retrieve item properties by themselves. Through Visual Studio .NET you can create your own ASPs to interact with your data. Those who are more familiar with programming and the functionality of Visual Studio .NET prefer the additional functionality of creating their own ASPs.

- **You can reuse Outlook Web Access forms and elements.** It is easy to develop frame–based Web pages by using FrontPage. When you set each frame to use a particular Web page, you may specify URLs that point to Web storage system folders or individual items. OWA then is used to render the contents in these frames. For instance, you can build a page that displays both the calendar and the in–box folder.

- **The forms registry should be used to reference available forms.** To use custom Web pages in the Web storage system, you need to reference them in registration items. Every Web storage system folder provides access to an urn:schemas–microsoft–com:exch–data:schema–collection–ref property that identifies the default URL of the folder in which to look for schema definition items and forms registries. If this property is not set, the default location is non_ipm_subtree/Schema in the folder's mailbox or public store. The forms registration is based on items with a DAV:contentclass of urn:schemas–microsoft–com:office:forms#registration. Various properties can be set for this item type to control how the referenced Web page is used. For instance, one Web page may represent the default view for a folder, while another may be launched when an item of a particular content class is opened. Data and forms must reside on the same server. Remote access to forms over the network is not supported. On the basis of the information obtained from the browser (that is, requested URL and HTTP header information), the Web storage system looks for a form definition that matches the requested item, the browser capabilities, and other information. Through a best–fit comparison, the appropriate form is determined for the rendering process.

Database Interfaces

The Web storage system provides an Exchange OLE DB (ExOLEDB) provider that supports record–level access to messaging information. Applications can use OLE DB and ADO 2.5 interfaces to communicate with ExOLEDB. If ADO is used, the ADO runtime dynamically linked libraries (DLLs) respond to the source application's request to open a record set by communicating with OLE DB. This OLE DB provider gives the developer direct access to data in a mailbox or public folder. This is beneficial in retrieving contact or calender information from a shared public folder. Otherwise, the call is passed to OLE DB runtime DLLs, which in both cases send the request to the OLE DB provider. ADO is installed with Windows 2000 Server. ExOLEDB comes with Exchange 2000 Server. Figure 7.4 diagrams the data access process for an Exchange information store.

Data Access to the Exchange Information Store
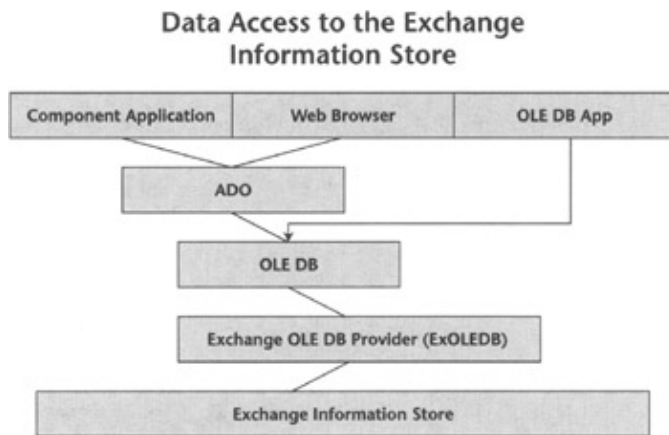
Figure 7.4: Application developers can use the Exchange OLE DB provider to access data stored in an Exchange information store.

ExOLEDB is a server–side component that is registered automatically during the installation of Exchange Server 2000. It is important to note that ExOLEDB can be used directly only on the local server. ASP.NET applications, ActiveX DLLs, and EXEs as well as event sinks running on the server may use this provider to access local resources and return the information to remote clients. To access remote resources, however, HTTP/WebDAV or MAPI must be used. ADO.NET provides access to the Web Storage System similar to Microsoft Access or Microsoft SQL Server. If you are familiar with developing SQL applications, you can easily write Web storage system applications by using the same development tools.

Exchange 2000 Server supports SQL commands. To open record sets to work with individual items, one can use the record object. Furthermore, ADO.NET supports a fields collection to access item properties and define custom properties. An ADO streams object provides direct access to an item's content.

A common use for the ExOLEDB is to create custom applications. In most cases developers choose to use a relational database management system such as SQL Server for the back end of client/server applications. This is generally a good idea. It is beneficial in some cases to use a product such as Exchange Server as the database back end for an application. You may want to consider using Exchange Server when the application you are developing is very dependent on the services provided by Exchange Server. Some of these services include the following:

- Email messaging
- Message routing
- Calender features

For example, if you have an application that naturally requires an approval process, Exchange Server may be a good solution. A purchase order system is a good example if multiple individuals need to approve a purchase order before it is paid. The message–routing features of Exchange Server can route the message automatically to the next individual in the process.

Through the ExOLEDB provider it is also much easier to transfer the pertinent data from Exchange Server to the larger database management systems. In the purchase order example, one may need the information tranferred to a receivables and accounting package. The DTS of SQL Server can transfer data from Exchange Server to SQL Server or another database that is storing accounting information.

For developers who use ADO.NET on a regular basis, interfacing with Exchange Server is a similar process. For information on using ADO.NET, refer to *Programming ADO.NET* by Richard

Hundhausen and Steven Borg (Wiley Publishing, Inc., 2002).

Exchange Installable File System

As was mentioned earlier, standard Win32 applications can be used to read and write files directly in the Web storage system. An ExIFS driver allows you to access message streams by using functions from Microsoft Win32 file Application Programming Interfaces (APIs) such as CreateFile, ReadFile, and WriteFile. Item properties are inaccessible, but they can be retrieved by using ExOLEDB, HTTP/WebDAV, or MAPI.

You can examine a message stream by opening an item via ExIFS in Notepad. Message streams are encoded in Multipurpose Internet Mail Extensions (MIMEs). The ExIFS can be used to share existing Exchange Server resources or allow Exchange Server to be used as a repository for other applications. The next two sections provide more details about each of these options. In many cases you would use the ExIFS to access or store data in Exchange Server and then use a Web interface (XML, HTTP, WebDAV, or XMLHTTP) to view the data.

Sharing Existing Resources

By default, Exchange 2000 Server maps the local M drive to the ExIFS. If M is already in use, the next available drive letter is taken. The ExIFS hierarchy starts with a folder that is named according to your Exchange 2000 Server organization, followed by sublevel objects called public folders and Mbx. If alternative public folder hierarchies have been created, they will be listed at this level as well.

In the same way that you can share ordinary directories on the server for access over the network, you can share Exchange 2000 Server resources. Unfortunately, these network shares are lost during a server reboot. ExIFS does not preserve the configuration, and this requires you to re–create the network shares after each server restart. A script can be used to simplify the task of re–creating network shares by using Net share commands. Because this has to be maintained, you may find it easier to access the data via HTTP or a database interface, as previously described.

Storage for Application Data

Direct file access through ExIFS makes it possible to store Web pages applications and then access them via a Web or data interface. In this way you can implement and deploy entire collaboration applications in the Web storage system. Rather than creating virtual directories on the server's local file system, you can create virtual directories in the Web storage system. Installing applications within the ExIFS provides the following advantages:

- Exchange System Manager can manage some pieces of the application from the server side. All application services can be integrated in a single location, making it easier to find, use, and share information.
- Exchange Server has built–in replication features to copy data to multiple servers.
- Exchange Server backup and restore features can be used for the data for your application.
- Collaboration applications can benefit from advanced system configurations such as .NET clustering.
- The Web storage system supports transaction logging and ensures data integrity through redundancy.

Integration with .NET Enterprise Servers

Exchange Server traditionally has not integrated well with the other servers in the Microsoft Enterprise Servers. With the changes to Exchange Server 2000, the data that are stored in Exchange Server are now accessible from the interfaces that are used frequently to access other database management systems. This allows messaging data that are stored in a mailbox store or public folder store to be accessed by developers and transferred to other applications. Exchange Server 2000 integrates with several of the other .NET Enterprise Servers. This section outlines the integration of Exchange Server with the other .NET Enterprise Servers by outlining the role of Exchange Server in enterprise application integration (EAI). It then identifies the role of Exchange Server in business–to–business (B2B) applications. Finally, it outlines the role of Exchange Server in business–to–consumer (B2C) applications.

Enterprise Application Integration

Integration is the strength of Exchange Server integration. Exchange Server typically is not used in B2B or B2C applications, although it is not precluded from participating in them. However, it is common for an organization to use Exchange Server in its EAI solution. This can be helpful from a couple of different perspectives. The first is the integration of traditional email information into other applications. The second is the integration of a custom–built application that stores its data and forms in Exchange Server with other enterprise–level applications. The following sections outline these options in more detail.

Traditionazl Email Information

As the role of messaging has evolved in corporate America, dependence on email has become more entrenched in people's everyday lives. Consequently, many users spend more time in the email application (typically Outlook in an Exchange Server environment) than in any other application. The add–on features to messaging have become critical. Many organizations now depend on the scheduling and calendar information in Outlook as much as they do on the email functionality. It is becoming more common for organizations to want this information to be accessible from more applications than the email product.

For instance, you may want your human resources package to store the scheduling information for the managers of your organization. In this example let us say the human resources package is stored in an SQL Server database. You could use the ExOLEDB provider to access the calander information for your managers and import the relevant data into the SQL Server database. You could schedule this process in an SQL Server Data Transformation Services (DTS) package. At that point you also could control how often you refresh the information, and because the amount of data retrieved is not large, the impact on your network will be relatively minimal. Then the human resources package could have the managers' scheduling and calendar information to help with the scheduling of interviews for new employees or evaluations for existing employees.

Custom Applications

The second option for EAI functionality in Exchange Server is with custom applications in Exchange Server. The first thing to be addressed is why one should develop the application within Exchange Server in the first place. Exchange Server has built in messaging and workflow features that make it easy to route a message or a request from one user to another or from one location to another. Exchange Server is an excellent development environment for applications or business processes that naturally need to flow from one location to another. The following applications are examples of message flow applications in Exchange Server:

- Purchase–order approval system
- Help desk request and solution–tracking system
- Customer request forms
- Product testing and review systems

These applications can be developed in Exchange Server. The data are stored in either a mailbox store or (typically) a public folder store. The front end of the application is a Web form, an Outlook form, or a custom application developed in a product such as Visual Basic .NET. The data from these applications may have to be integrated with other applications in the organization. You can use XML or ADO.NET to retrieve the data and make them available to other applications.

For example, let us say the organization has designed a purchase–order approval system that is totally Web–based. The system was created with ASP.NET, and the data and Web pages are stored in IIS and Exchange Server. You are storing the purchase orders and their status in a public folder store. You chose to develop the application in Exchange Server because the natural process of your purchase orders requires the routing of the purchase order to two managers for approval. The message–routing functions of Exchange Server help automate the delivery of the requests to the appropriate manager for approval. You also have an accounting package. That package uses a Visual Basic .NET front end that stores the data in an SQL Server database. You would like the accounting package to query the purchase–order status from Exchange Server and update the accounts payable system after the purchase order has been approved. In this scenario you could use either XML or ADO.NET through the ExOLEDB provider to query the status and amount of the purchase orders in the Exchange Server application. You could then use XML or ADO.NET to write the updated information to the SQL Server database. Your accounting application then would reflect the status of your purchase orders. In this example you are not transferring the data from one server to another; you are using both applications together to maintain the integrity of the data in both applications.

As can be seen, the interfaces that are available with Exchange Server 2000 assist in the integration of applications within an organization. Exchange Server can now play a significant role in the application development arena within a company.

Business–to–Business Applications

Exchange is not used commonly in B2B applications. It can, however, play a role in the integration of two organizations that are highly dependent on each other.

For example, let us expand the purchase–order example. The purchase–order application was created to route the approval of purchase orders from the requester to the approving managers. Let us extend this to start with the vendor that is providing the service. The vendor could use the application to present the purchase order to your organization for approval. This could be accomplished through the use of a couple of different options. The first option is to provide the vendor with access to the Web forms that are used to input the purchase–order information. The vendor could then enter the information directly and wait for its money. This requires the vendor to enter the data manually into your system.

However, in most cases the vendor also has an application where the original invoice was entered into its system, and so you could instead use BizTalk . Server to pull the data out of the vendor's systems by using XML and transfer the data to your Exchange Server purchase–order system. You can use BizTalk Server to access the data and define how the data should be mapped from one system to the other. This is where you make all your data transformations. This is an effective method for transferring data from any system that can expose the data via XML. The advantage of

this model is that you have decreased duplicate data entry and moved closer to a paperless solution. You may not want to give this type of access to all your vendors and they may not want to give it to you, but this is a highly effective solution for organizations that are tightly integrated.

Business–to–Consumer Solutions

Although Exchange Server can be used for B2C solutions, it is probably not the best choice for these types of applications. Commerce Server and Content Management Server were created to service customers more effectively. You should look to them before you start developing all your customer–related applications within Exchange Server 2000. You could store the ASP.NET pages that were used for your Web–based application within an Exchange Server public folder. The Web users then could access the data stored in your Exchange Server system. This is not common, and in most cases the ASP.NET pages your customers interact with should be stored on a Web Server running IIS that is accessible from the Internet.

Review Questions

1. What Internet protocols are supported by Exchange Server 2000?
2. What interfaces are available for accessing the Exchange Server information store?
3. What is the purpose of the Web storage system?
4. What is the Exchange Installable File System?
5. What is XMLHTTP?
6. What other .NET Enterprise Servers commonly interact with Exchange Server 2000?

Chapter 8: SQL Server 2000

Overview

SQL Server 2000 is a complete database and analysis solution that provides rapid delivery of the next generation of scalable Web applications. Structured Query Language (SQL) Server 2000 is a key component in supporting e−commerce, line−of−business, and data−warehousing applications while offering the scalability needed to support growing organizations. SQL Server 2000 takes full advantage of .NET Enterprise Servers by integrating with Active Directory Services and supporting up to 32 processors and 64 gigabytes (GB) of random access memory (RAM).

SQL Server 2000 is the basis for many .NET enterprise solutions. Many of the other .NET Enterprise Servers are dependent on SQL Server for the storage and retrieval of their data. SQL Server is the foundation for these applications.

SQL Server 2000 is scalable to levels of performance that can handle large Internet sites. In addition, the SQL Server 2000 database engine includes native support for XML, and the Web Assistant wizard helps generate HTML pages from SQL Server 2000 data and post SQL Server 2000 data to HyperText Transport Protocol (HTTP) and File Transfer Protocol (FTP) locations.

SQL Server supports Windows authentication, which enables .NET Server domain accounts to be used as SQL Server 2000 login accounts. Users are validated by .NET Active Directory services when they connect to the network. When a connection is formed with SQL Server, the SQL Server client software requests a trusted connection, which can be granted only if it has been validated by .NET Server. Thus, SQL Server does not have to validate users separately.

This chapter describes the different editions of SQL Server to assist developers in deciding which version to implement. The chapter then introduces the features of SQL Server that provide the foundation for .NET solutions. These features include data and retrieval features as well as data protection and scalability. This product is a focal point for many of the other products in the .NET Server suite, and understanding the development features of the product can help a developer create applications that need to scale and use SQL Server as the back end. The chapter ends with an explanation of how SQL Server interacts with some of the other .NET Enterprise Servers to create enterprise solutions.

SQL Server Editions

SQL Server 2000 is available in different editions to accommodate the unique performance, runtime, and price requirements of different organizations and individuals. The Enterprise Edition has some significant advantages over the Standard Edition. These features, however, generally enhance the scalability and stability of the SQL Server. The additional features are not always required, and you should consider that when you purchase the software. It is important to know the features of the edition you are using. You do not want to have overkill by purchasing the Enterprise Edition if it is not needed. However, many of the features that are available only in the Enterprise Edition are the focus of this chapter. The additional features provided with the Enterprise Edition facilitate enterprise development.

- **SQL Server 2000 Enterprise Edition.** This edition is the complete SQL Server offering for any organization. It offers the advanced scalability and reliability features that are necessary for mission−critical line−of−business and Internet scenarios, including distributed partitioned

views, log shipping, and enhanced failover clustering. This edition also takes full advantage of the highest–end hardware, with support for up to 32 central processing units (CPUs) and 64 GB of RAM. In addition, the Enterprise Edition includes advanced analysis features. Ultimately, the purchaser needs to determine whether the additional features listed here are required for his or her deployment of SQL Server.

- **SQL Server 2000 Standard Edition.** This edition is the affordable option for small and medium–size organizations that do not require advanced scalability and availability features or all the more advanced analysis features of the Enterprise Edition. Standard Edition can be used on symmetric multiprocessing systems with up to 4 CPUs and 2 GB of RAM. This is becoming a limitation for large organizations with multiple instances of SQL Server running on a single machine.
- **SQL Server 2000 Personal Edition.** This edition includes a full set of management tools and most of the functionality of the Standard Edition, but it is optimized for personal use. In addition to running on Microsoft's server operating systems, the Personal Edition runs on nonserver operating systems, including Windows 2000 Professional, Windows NT Workstation 4.0, and Windows 98. Dual–processor systems also are supported. While this edition supports databases of any size, its performance is optimized for single users and small workgroups and degrades with workloads generated by more than five concurrent users.
- **SQL Server 2000 Developer Edition.** The 2000 Developer Edition of SQL Server enables developers to build any type of application on top of SQL Server. This edition includes all the functionality of the Enterprise Edition, but with a special development and test end–user license agreement (EULA) that prohibits production deployment. For those who do not require end–user connectivity, this server is the best choice. This is often the case for a development server. You do not have to buy the Enterprise Edition of SQL Server for your development server.
- **SQL Server 2000 Desktop Engine (MSDE).** This edition has the basic database engine features of SQL Server 2000 but does not include a user interface, management tools, analysis capabilities, merge replication support, client access licenses, developer libraries, or Books Online. It also limits the database size and user workload. Desktop Engine has the smallest footprint of any edition of SQL Server 2000 and is thus an ideal embedded or offline data store.
- **SQL Server 2000 Windows CE Edition.** The 2000 Windows CE Edition is the version of SQL Server 2000 for devices and appliances that run Windows CE. This edition is programmatically compatible with the other editions of SQL Server 2000, and so developers can leverage their existing skills and applications to extend the power of a relational data store to solutions that run on new classes of devices.

.NET Enterprise Features

Several of the new features of SQL Server 2000 help with the deployment of scalable solutions throughout an organization. SQL Server 2000 has been enhanced to better support .NET solutions. This section introduces the new features of SQL Server that facilitate interaction with .NET. These features fall into two important categories: features that facilitate the retrieval and storage of data and features that ensure that the data are protected. These two issues are discussed separately in this chapter.

Protecting the data involves three goals. SQL Server 2000 includes several new features that help ensure that the following goals are met:

- The database should scale as the data grow without degrading performance significantly.

- The database application has to be secure. This involves not only user authentication but also isolation from other applications running on the same server.
- The database must be available. When the front−end application requests data, the server should be available to process the request.

Data Retrieval and Storage

SQL Server is a relational database product. Its primary goal is to store data and make the data available when properly requested. Several .NET Enterprise Servers connect to SQL Server to retrieve data. After the data are retrieved, they can be exported to another location or presented to a user through an application.

For example, Commerce Server can create secure Web sites. The products that are presented to the Web users at a site are stored in an SQL Server database. After a user has authenticated at the Web site, Commerce Server connects to SQL Server to retrieve the product list that should be displayed to the Web user. SQL Server plays the role of a storage mechanism.

Additionally, many of the .NET Enterprise Servers, such as Commerce Server (see Chapter 10, "Commerce Server") and BizTalk Server (see Chapter 12, "BizTalk Server"), store their configuration information in SQL Server databases. SQL Server 2000 is a requirement for many of the other .NET Enterprise Servers. The data are the core of all application processes. SQL Server 2000 is the foundation for many .NET Enterprise solutions.

This section describes the features that facilitate data storage and retrieval. It begins with an overview to the database engine, including references to the traditional methods of accessing data from a database, such as ADO (ActiveX Data Objects), OLE DB, and ODBC (Open Database Connectivity). Many .NET Enterprise Servers use these interfaces to access SQL Server.

This section then addresses XML, the new feature for data access and retrieval with SQL Server. XML and its interaction with SQL Server are this section's primary concern. After introducing XML and SQL Server 2000 interaction, the section ends with a description of the SQL Server Web Services Toolkit.

SQL Server 2000 Relational Database Engine

The SQL Server 2000 relational database engine is a highly scalable engine for storing data. The database engine stores data in tables. An application submits an SQL statement to the database engine, which returns the result to the application in the form of a tabular result set. An Internet application submits either an SQL statement or an XPATH query to the database engine, which returns the result as an XML document. The relational database engine provides support for the common Microsoft data access interfaces, such as ADO, OLE DB, and ODBC. With Data Transformation Services (DTS) it is possible to transfer data to and from SQL Server databases by using OLE DB.

DTS has been available in SQL Server since Version 7.0. DTS is a tool for the movement and transformation of data from one location to another. The advantage of DTS is the direct access to any data source that it provides through OLE DB. This results in a fast connection.

The downside to DTS is that it does not provide for a shared application logic. When you transfer data from one data source to another, you must be comfortable with the database schemas of both locations. You also have to build transformations to map data between the two systems. If you are comfortable with the schema at both locations, this may not be a difficult process; if you are not, this

process may not seem possible.

XML and SQL Server 2000

Both XML and HTML are derived from Standard Generalized Markup Language (SGML). SGML is a large, complex language that is difficult to use for publishing data on the Web. HTML is a simpler, specialized markup language but has a number of limitations when one is working with data on the Web.

Once the information is transferred to HTML, the data lose their structure because everything is stored as text. With HTML there is no equivalent to data typing, and this makes it very limited when one is working with databases. XML is smaller than SGML and more robust than HTML, and so it is becoming an increasingly important language in the exchange of electronic data through the Web. XML allows the maintenance of data structure while the information is transferred to and from the Web.

In a relational database such as SQL Server, the result set of a SELECT statement is in the form of a table. Traditional client/server applications that execute a SELECT statement process the result by fetching one row or a block of rows at a time from the tabular result set and mapping the column values into program variables.

SQL Server 2000 includes many features that support XML's functionality. The combination of these features makes SQL Server 2000 an XML–enabled database server. The following features in SQL Server 2000 support XML functionality:

- The ability to access SQL Server through HTTP. See *Accessing SQL Server by Using HTTP* later in this chapter for further information.
- The ability to retrieve data by using the SELECT statement and the FOR XML clause.
- The ability to retrieve data by using the XPATH query language.
- The ability to write XML data to SQL Server by using the OPENXML rowset provider.
- Support for XML–Data Reduced (XDR) schemas and the ability to specify XPATH queries against those schemas.
- Enhancements to the SQL Server OLE DB (SQLOLEDB) provider that enable XML documents to be set as command text and return result sets as a stream.

Note   XML functionality is sophisticated and advanced. This chapter only touches the surface of XML and focuses on its integration with SQL Server. For more information, refer to *Chapter 5*, "XML Integration with .NET Servers."

XML support in SQL Server is not difficult to configure. There are two primary considerations, which are addressed in the next two sections: configuring Internet Information Server (IIS) for XML support in SQL Server and querying the data by using HTTP.

Configuring SQL Server Support In Internet Information Server

Before accessing a SQL Server 2000 database via HTTP, it is necessary to set up an appropriate virtual directory. You can use the IIS Virtual Directory Management for SQL Server utility to define and register a new virtual directory, also known as the virtual root, on the computer running Microsoft IIS. This utility instructs IIS to create an association between the new virtual directory and an instance of Microsoft SQL Server. When you create the virtual root, you need to be able to supply the following items:

- The name of the server you want to publish in IIS.

- The name of the virtual directory. Queries will have to reference the server name and the virtual directory name to access the data.
- The security method that will be used for the connection. You need to know whether you will use Windows authentication or SQL Server authentication.
- The path where the database files is stored.
- The options you want to support. You can configure the virtual root to support HTTP Uniform Resource Locator (URL) queries, template queries, XPATH queries, and POST operations. URL queries allow users to pass a query in the URL. Template queries allow users to execute queries stored in an XML template. XPATH queries allow users to execute XPATH queries over SQL Views. XPATH allows a developer to create XML views of relational data and then query them. POST allows the user to post an XML template to the virtual directory.

After you have made the decisions about all the information you have to supply, you can create the virtual directory. To create a new virtual root for your instance of SQL Server 2000, perform the following steps:

1. From the SQL Server program group click Configure XML Support in IIS to open the IIS Virtual Management for SQL Server utility, as shown in Figure 8.1.
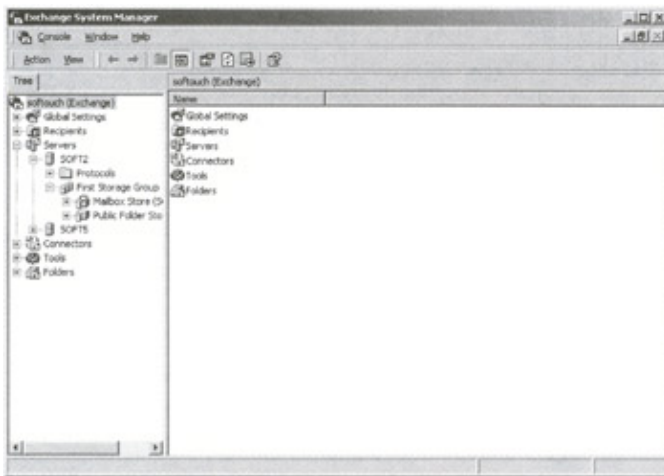
Figure 8.1: The IIS Virtual Management for SQL Server utility can configure HTTP support for SQL Server.

2. Click to expand the server name.
3. Right−click the Web site where you want to create the virtual directory and select New Virtual Directory.
4. From the General tab of the New Virtual Directory Properties dialogue box, specify the name of the virtual directory and the path to your database files.
5. Click the Security tab to choose the authentication method.
6. Click the Data Source tab to define the server for the virtual directory.
7. Click the Settings tab to define the query options you want to support, as shown in Figure 8.2.
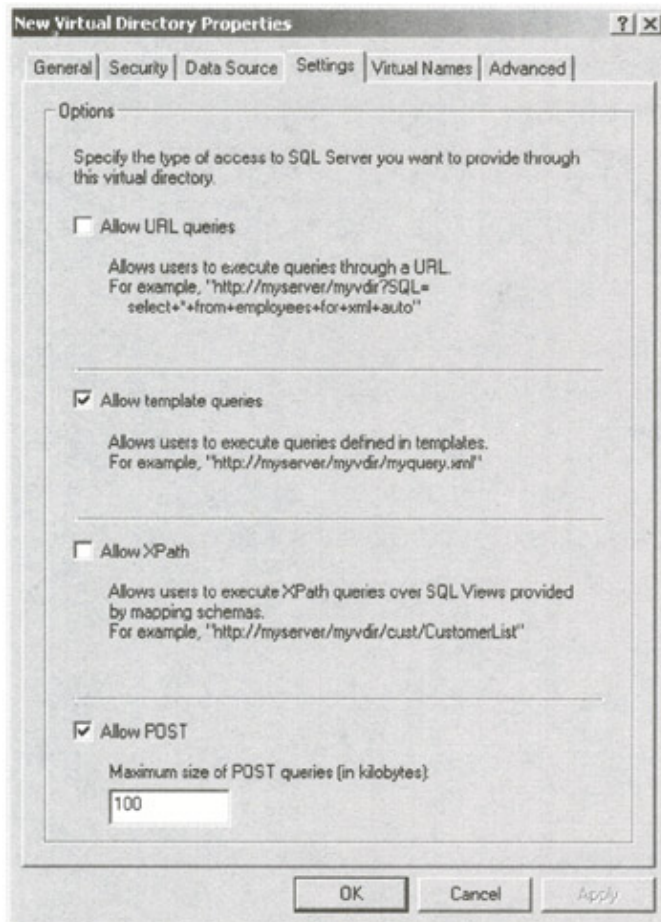
Figure 8.2: URL queries, XPATH queries, and template queries can be allowed after XML support has been configured for SQL Server.

8. Click OK to create the virtual directory.

Accessing SQL Server by Using HTTP

It is possible to access SQL Server 2000 by using HTTP. The HTTP access to SQL Server allows you to access data through the following types of queries:

- Specify SQL queries directly in the URL. For example, you could type the following in a URL request:

```
http://Chicago/nwind?squl=SELECT+*+FROM+Customers+FOR+XML+AUTO&root=root
```

The FOR XML clause in this example returns the result as an XML document instead of a standard rowset. The root parameter identifies the single top–level element. One can specify templates directly in the URL. Templates are valid XML documents that contain one or more SQL statements. They allow you to put together data to form a valid XML document; this is not necessarily the case when queries are specified directly in the URL. For example, you could use the following to retrieve data using a template query:

```
http://Chicago/nwind?template=<ROOT+xmlns:sql="urn:schemas-microsoft-
com:xml-squl"><sql:query>SELECT+*+FROM+Customers+FOR+XML+AUTO</sql:qu
ery></ROOT>
```

- Specify template files in the URL. Writing long SQL queries at the URL can be cumbersome. In addition, browsers may have limitations in regard to the amount of text that can be entered in the URL. To avoid these problems, templates can be written and stored in a file. A

template is a valid XML document containing one or more SQL statements and XPATH queries. You can specify a template file directly in a URL by typing the following:

```
http://Chicago/nwind/TemplateVirtualName/templatefile.xml
```

In this example TemplateVirtualName is the virtual name of the template type that is created by using the IIS Virtual Directory Management for SQL Server utility. Template files also enhance security by removing the details of database queries from the user. When the template file is stored in the virtual root directory (or its subdirectories) where the database is registered, security can be enforced by removing the URL query–processing service on the virtual root and leaving only the SQL Server XML Internet Server Application Programming Interface (ISAPI) to process the files and return the result set.

• Write XPATH queries against the annotated XDR schemas (also referred to as mapping schemas). Writing XPATH queries against the mapping schemas is conceptually similar to creating views by using the CREATE VIEW statement and writing SQL queries against them by typing the following:

```
http://Chicago/nwind/SchemaVirtualName/schemafile.xml/Customer[@Custo
merID="ALFKI"]
```

In this example SchemaVirtualName is the virtual name of the schema type that is created by using the IIS Virtual Directory Management for SQL Server utility. Customer[@CustomerID="ALFKI"] is the XPATH query executed against the schemafile.xml specified in the URL.

• Specify database objects directly in the URL. The database objects, such as tables and views, can be specified as parts of the URL, and an XPATH can be specified against the database object by typing the following:

```
http://Chicago/nwind/dbobjectVirtualName/ XPATH Query
```

In this example dbobjectVirtualName is the virtual name of the dbobject type that is created by using the IIS Virtual Directory Management for SQL Server utility.

SQL Server Web Services Toolkit

Data that are in XML format can be read on any platform that supports XML. Most data are still stored in a relational database that all the current applications can use effectively. With the release of SQL Server 2000, exposing data as XML becomes possible without extensive transformations and program modules. Now current applications can continue to access the data with the current interfaces, such as ADO, and it is possible to expose data as XML views to other platforms that support XML. Over time, as the W3C standards for XML have evolved, SQLXML (XML for SQL Server) has continued to build on and expand its functionality by including XML Schema (XSD) support, updategrams, and integrated Microsoft .NET support.

The SQL Server 2000 Web Services Toolkit delivers tools, code, samples, and white papers for building XML Web services and Web applications with SQL Server 2000. This enables developers to create XML Web services via SQLXML 3.0 easily.

SQLXML 3.0 extends the built–in XML capabilities of SQL Server 2000 with technology that allows the creation of XML Web services from SQL Server stored procedures or server–side XML templates. SQLXML 3.0 also includes extensions to the .NET Framework that provide SQLXML programmability to the languages supported by Visual Studio .NET, including C# and Microsoft Visual Basic .NET. The Microsoft SQL Server 2000 Web Services Toolkit is available for

downloading at the Microsoft Web site; it can be found in MSDN under the .NET Enterprise Servers. At the time of this writing the URL for the toolkit is www.microsoft.com/ServiceProviders/downloads/sql_tlkit_pll5956.asp.

The Web Services Toolkit provides the following advantages to developers:

- **SQLXML version 3.0.** SQL Server 2000 is shipped with support for storing and retrieving data via XML, as was outlined in the previous section. To keep up with growing and changing standards for XML, Microsoft began releasing feature packs for XML. These packs are available as free downloads and are called SQLXML.
- **Support for Web services.** SQLXML 3.0 introduces Web services in SQL Server 2000. With SQLXML 3.0, stored procedures and XML templates can be exposed as Web services via SOAP. Thus, they can be accessed from the Web in the same manner as any other Web service. For more information on Web Services, refer to Chapter 15, "Web Services with .NET."
- **Support for the SQLXML OLE DB provider.** The SQLOLEDB provider can be used with XML to retrieve data from a database. The SQLOLEDB provider shapes the data for XML, whereas the SQLXML provider streams the data from SQL Server. If you are using ADO and SQLXML, you should always use the SQLXMLOLEDB provider to query your database for the data. Using the SQLXMLOLEDB Provider is faster than using the SQLOLEDB. Currently, the provider is available only for ADO, and so there is no access to it from other SQL Server features, such as DTS and linked servers. For your connections to use the SQLXMLOLEDB provider, enter the provider in the connection string, as shown in the following example:

```
con.open "Provider=SQLXMLOLEDB:data provider=sqloledb;integrated
security=true;server=(local);database=northwind"
```
- **SQLXML managed classes.** If the .NET Framework is installed on your machine, the SQLXML managed classes allow you to author .NET code to perform functions that previously were available only through SQLXML. You then can perform these functions from the language and development environment of your choice. With SQLXML managed classes you can retrieve XML from an SQL database or run XML on the client through stored procedures, ad hoc queries, XML templates, or annotated schemas with XPATH. For a detailed outline of the object model and instructions on how to interact with each class, download the toolkit, which has several white papers, including one that outlines the SQLXML managed classes.

Data Protection

Data retrieval and storage are important, but because SQL Server becomes the core of your business process, the protection of those data is equally important. .NET Enterprise solutions should be able to do the following:

- **Grow as the data needs of an application grow.** The growth should not decrease the performance of the application significantly.
- **Ensure security by user and by application.** Security should be managed not only by user but also by application. Only securely authenticated users should have access to the data, and the application should be isolated from the other applications that are running on the server. If one application crashes, you do not want it to bring down all the applications.
- **Provide a high level of availability.** It is dangerous to claim 100 percent availability, but everyone wants that. You should be able to provide a solution that gets as close as possible to 100 percent availability.

This section describes the SQL Server 2000 features that help a developer meet the requirements for enterprise solutions described above. This section includes details about the following features of SQL Server 2000. Some of the most significant enhancements to SQL Server in the 2000 version have come in the areas of security and availability, which are discussed separately under the following headings:

- Federated database Servers
- Windows authentication
- Multiple instances of SQL Server on a single machine
- Clustering services

Federated Database Servers

A primary concern for many enterprise solutions is the ability to scale an application as data requirements grow. Microsoft SQL Server 2000 databases can be spread across a group of database servers. This provides a database solution that can support the processing growth requirements of the largest Web sites and enterprise data–processing systems.

SQL Server 2000 supports updatable distributed partitioned views that are used to transparently partition data horizontally across a group of servers. The servers involved in the distributed view are referred to as federated database servers. Although these servers cooperate in managing partitioned data, they operate and are managed separately.

While SQL Server 2000 delivers impressive performance when scaled on servers with eight or more processors, it can support huge processing loads when partitioned across a federation. The federation is dependent on all the machines involved working together to act as a single database. They need similar security settings to make sure users can interact with all the machines in the federation. Although the user sees one view of the data, security is configured separately for each server. This process is easy to configure if the user setting up the federation is a system administrator.

A federated database tier can achieve extremely high levels of performance only if the application sends each SQL statement to the member server that has most of the data required by a statement. This is called collocating the SQL statement with the data required by that statement. Collocating SQL statements with the required data is not a requirement unique to federated servers; it also is required in clustered systems (see Chapter 18, "Clustering .NET Servers").

Although a federation of servers presents the same image to applications that a single database server presents, there are internal differences in how the database services tier is implemented. Table 8.1 identifies the differences between a single server application and a federated server tiered application.

Table 8.1: Single Server Applications versus Federated Server Applications

SINGLE SERVER	FEDERATED SERVERS
Only one instance of SQL Server needed.	One instance required for each member of the federation.
Production data are stored physically in a single database.	Each member has a database, and the data are spread across the servers.
Each table is singular.	The table from the original database is partitioned horizontally into tables on each of the member servers. Distributed partitioned views

	make it appear as though the data were in a single location.
Connection is made to a single server.	The application layer must collocate SQL statements to ensure that the server that has most of the data receives the request.

While the goal is to design a federation of database servers to handle a complete workload, this is done by designing a set of distributed partitioned views that spread the data across the different servers. If the design is not solid in the beginning, the performance of the queries will suffer as the servers try to build the result sets required by the queries.

This section discusses the details of configuring the distributed partitioned view. Then the considerations for updating, inserting, and deleting data are introduced. Finally, the section addresses the security concerns related to federated database servers.

Creating a Partitioned View

A partitioned view joins horizontally partitioned data from a set of member tables across one or more servers, making the data appear as if it came from one table. SQL Server recognizes the difference between local views and distributed partitioned views. In a local partitioned view, all participating tables and the view reside on the same instance of SQL Server. In a distributed partitioned view, at least one of the participating tables resides on a different server.

SQL Server also differentiates between partitioned views that are updatable and views that are read–only copies of the underlying tables. Although the view that is created may be updatable, the user that interacts with the view still must be given permission to update the distributed view. The permissions for these partitioned views are similar to those for regular views, but it is necessary to configure the permission on each server referenced in the partitioned view.

Before implementing a partitioned view, you must partition a table horizontally. The original table is replaced with several smaller member tables. Each member table has the same number of columns and the same configuration as the original table. If you are creating a distributed partitioned view, each member table is stored on a separate member server. The name of the member databases should be the same on each member server. This is not a requirement, but it helps decrease confusion.

The member tables are designed so that each table stores a horizontal slice of the original table based on a range of key values. The ranges are based on the data values in a partitioning column. The range of values in each member table is enforced by a CHECK constraint on the partitioning column, and ranges cannot overlap. For example, you cannot have one table with a range from 1 through 200,000 and another with a range from 150,000 through 300,000 because then it will not be clear which table contains the values from 150,000 through 200,000. If you are partitioning a customer table into three tables, the CHECK constraint for these tables could appear as follows:

```
-- On Server1:
CREATE TABLE Customer_33
  (CustomerID   INTEGER PRIMARY KEY
             CHECK (CustomerID BETWEEN 1 AND 32999),
  ... -- Additional column definitions)
-- On Server2:
CREATE TABLE Customer_66
  (CustomerID   INTEGER PRIMARY KEY
             CHECK (CustomerID BETWEEN 33000 AND 65999),
  ... -- Additional column definitions)
-- On Server3:
CREATE TABLE Customer_99
```

```
(CustomerID   INTEGER PRIMARY KEY
              CHECK (CustomerID BETWEEN 66000 AND 99999),
... -- Additional column definitions)
```

Note You need permission to create tables on all the servers involved in a federation.

After creating the member tables, you define a distributed partitioned view on each member server. The view names on all thee servers should be the same. This allows queries referencing the distributed partitioned view name to run on any of the member servers. The system operates as if a copy of the original table were on each member server, but each server has only a member table and a distributed partitioned view. The location of the data is transparent to the application.

Creating distributed partitioned views requires several steps, which must be configured. To perform the necessary configuration options, perform the following steps:

1. Each member server has to be configured as a linked server on all the other member servers. This is necessary to allow every server to run a query and access the other servers when necessary. The security settings of the linked servers must be configured to allow all users to authenticate against all servers.
2. Set the lazy schema validation option, using sp_serveroption, for each linked server definition used in distributed partitioned views. This tells the query optimizer not to request metadata from the remote table until it actually needs data from the remote table. This optimizes the execution of the query and may prevent unnecessary retrieval of metadata. You need to be a member of the system administrator's role to set this value.
3. Create a distributed partitioned view on each member server. The views use distributed SELECT statements to access data from the linked member servers and merge the distributed rows with rows from the local member table. To complete this step, you must have permission to create views on all the servers. The following example demonstrates a distributed partitioned view. The SELECT statement must be performed against all the servers involved in the federation:

```
CREATE VIEW Customers AS
    SELECT * FROM CompanyDatabase.TableOwner.Customers_33
UNION ALL
    SELECT * FROM Server2.CompanyDatabase.TableOwner.Customers_66
UNION ALL
    SELECT * FROM Server3.CompanyDatabase.TableOwner.Customers_99
```

Distributed Partition View Rules

In addition to the rules defined for partitioned views, distributed partition views have the following additional conditions:

• A distributed transaction will be started to ensure atomicity across all the nodes affected by the update.
• The XACT_ABORT SET option must be set to ON.
• Smallmoney and smalldatetime columns in remote tables are mapped as money and datetime, respectively. Consequently, the corresponding columns in the local tables should be money and datetime.
• Any linked server cannot be a loopback linked server, that is, a linked server that points to the same instance of SQL Server.
• A view that references partitioned tables without following all these rules may still be updatable if there is an INSTEAD OF trigger on the view. The query optimizer, however, may not always be able to build execution plans for a view with an INSTEAD OF trigger that are as efficient as the plans for a partitioned view that follow all the rules.

Windows Authentication Mode

SQL Server 2000 provides the option to authenticate users through Windows authentication or through the use of SQL Server usernames and passwords.

In Windows authentication mode, only Windows users can be given access to SQL Server. Each user that accesses the SQL Server will be required to first log on to the Windows domain. The SA account is disabled in this mode, and SQL accounts cannot be created. The local administrators group from the local machine will be given access to SQL Server automatically, and all other Windows users and groups will have to be granted or denied access manually. Using Windows authentication mode allows access of the following Windows domain features for all users:

- **Encrypted authentication.** Usernames and passwords are encrypted. With SQL Server, authentication information is sent in clear text.
- **Password expiration, security, and other password policies.**
- **Group management for access to SQL Server.**
- **Single account management.** Passwords have to be maintained only at the domain level.

A connection to SQL Server made via Windows authentication often is referred to as a trusted connection. From your development tool, the connection to SQL Server is coded differently as a trusted connection than it is as a standard, nontrusted connection. The following is an example of a connection string using a trusted connection through ADO.NET. The security reference is highlighted in bold:

```
dim cn
set cn = Server.CreateObject("ADODB.Connection") 'creates connection
object
cn.Provider = "SQLOLEDB"  ' native SQL Server provider identifier
cn.ConnectionString =      "Data Source=ServerName;" & _
    "Initial Catalog=DatabaseName;" & _
"Integrated Security-SSPI;"
cn.Open
```

Multiple Instances of SQL Server

Multiple instances of SQL Server can be installed on a single server. In many organizations this is one of the best tools for isolating applications and guaranteeing a secure solution. You can set up one application with its database stored in one instance of SQL Server and another application with its database stored in another instance of SQL Server. Say you have two Web sites that need to be Web–enabled. One is used for employees to enter their time for the payroll system, and the other application stores the company credit card usage and approval system. Neither of these applications may require a tremendous amount of overhead, but you want them to be available when the employees have to enter their information. By storing the databases in two separate instances of SQL Server 2000, you can isolate the two applications from each other. If one of the databases fails, it is unlikely that the other will be affected by the outage.

To install multiple instances of SQL Server on a single machine, you need to run the setup program multiple times. You are prompted during the installation for the designation of the instance, as shown in Figure 8.3.
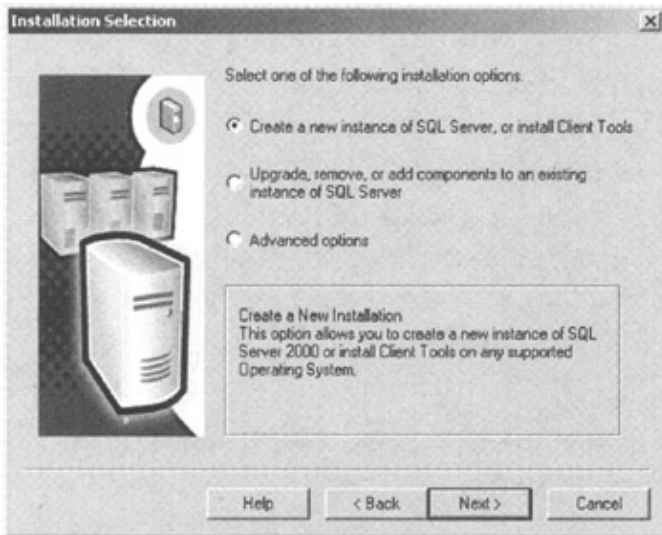
Figure 8.3: During the installation you are asked if you want to install a named instance of SQL Server.

If you choose to install another instance of SQL Server, you are prompted to name the instance, as shown in Figure 8.4. The name of the instance is appended to the name of the server to create the full name. For instance, if you installed an instance of SQL Server named NEW YORK on a computer named CHICAGO, the full name of the instance would be CHICAGO\NEWYORK. Additionally, clients that access the instance must have Microsoft Data Access Components (MDAC) 2.6 installed or the access to named instances will fail. MDAC 2.6 allows the data access components to recognize the \ symbol and include it in the path to the server instance. MDAC generally is installed with a Microsoft Office installation.
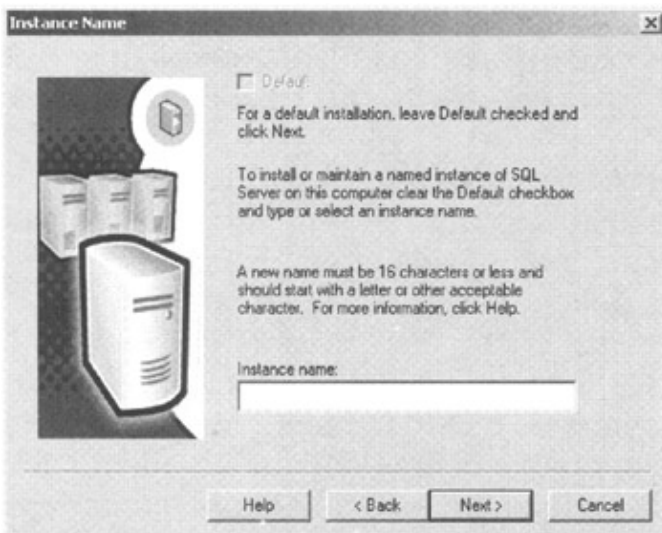
Figure 8.4: During the installation you are prompted for the name of the SQL Server instance.

> **Note**   When you install multiple instances of SQL Server, you must stop the services of the instances that already have been installed. It is best to perform the installation of subsequent instances during off hours.

By installing multiple instances of SQL Server you gain the following security and stability benefits:

- **Each instance has its own services.** If one application or instance fails, it will affect only its own services. This isolates the application and helps prevent it from harming another application.
- **Each instance has its own sysadmin role.** The system administrators for each instance are defined separately. This is very advantageous in a development environment where

database developers are given system administrator priviliges to their instances. This allows you to separate the developers and not let them affect one another's work.

- **Each instance has its own logins.** You can now add only the logins that are appropriate for a specific instance. If the instance is supporting one application that needs only four logins, you have to create only the four logins.

Note Applications that have known stability problems should be run on their own instance of SQL Server. This will isolate the application so that it does not affect the other applications.

Clustering Services

A cluster is a group of computers that work together to run a common set of applications and provide the appearance of a single point of contact. SQL Server 2000 can be installed on a .NET clustering solution. To the user, all the servers configured in the cluster are a single entity. The computers are physically connected by cables and programmatically connected by cluster software. These connections allow computers to use failover and load balancing, something that is not possible with a stand–alone computer. .NET Server clustering technology provides the following list of advantages that can be used to improve the stability of the application:

- **High availability.** The cluster is designed to avoid a single point of failure. Applications can be distributed over more than one computer, achieving a degree of parallelism and failure recovery and providing more availability.
- **Scalability.** The cluster's computing power can be increased by adding more computers.
- **Manageability.** The cluster appears as a single–system image to end users, applications, and the network, while providing a single point of control to administrators. The single point of control can be remote.

The first step in planning an SQL Server cluster is determining the type of hardware to use and the mode of operation in which the cluster will run. You have to choose whether the implementation of cluster service for SQL Server will be active/active or active/passive.

- Active/passive cluster configurations should consist of computers with identical hardware. Each of these machines should be capable of handling the workload of the databases. Because the active/passive mode does not use the secondary system during normal operations, the performance of the SQL Server instance should remain constant. Users will not experience any performance change if the primary system fails over to an identical secondary system. With an active/passive cluster you have made the choice to invest in the additional hardware to guarantee the performance of the application. The only performance hit will be the time it takes for the failover process. In most cases this should not exceed 90 seconds.
- Active/active cluster configurations should consist of two systems, each of which is running a specific workload. The configuration and performance will be optimal if both systems are identical from a hardware standpoint. If a failure occurs, the surviving system will take over the workload of the failed system. This will cause the load of both servers to run on one server. If either server was at a high utilization rate before the failover, this process most likely will result in decreased performance. With active/active clustering you typically are choosing to have a fault–tolerant solution at the expense of performance. When the system performs a failover, the performance may be slower but all the data will be available to the users. You generally should document the configuration of the cluster and inform users and other support staff that a failover will result in a performance decrease.

There are more details that can help you customize your cluster configuration for SQL Server. More information about the details of installing clustering and configuring it for SQL Server and Exchange

Server appear in Chapter 18, "Clustering .NET Servers."

Enterprise Solutions with SQL Server 2000

As the foundation of many business processes, SQL Server 2000 provides a core service for many of the .NET Enterprise Servers. Not only do the servers use SQL Server to retrieve the data needed for the application, many of the servers also use SQL Server to store their basic configuration information. For example, if you are creating a commerce site, you probably are using Commerce Server and Internet Information Server to present the site to users and authenticate their requests. When a Web user accesses the commerce site, Commerce Server must authenticate the user and then present the product listing that is available to the user. Commerce Server stores the product information in an SQL Server database. As the Web user continues to use the site, the user may change some personal preferences. As the preferences of the user change, the changes are stored as part of the user's profile. Commerce Server stores all profile–related information in an SQL Server database. SQL Server provides similar services for some of the other .NET Enterprise Servers.

This section demonstrates the role of SQL Server in common enterprise solutions. First there is an example of SQL Server's role in enterprise application integration (EAI), and then an example is provided for using SQL Server in a business–to–business model.

In a business–to–consumer (B2C) model you are exposing data to your customers. SQL Server is usually the repository for the data that are presented to a Web user. In most cases, the commerce site is stored outside the primary firewall and the SQL Server is stored behind the firewall. This process provides an additional security level. This model is described in more detail in other chapters. Chapter 9, "Application Center 2000," and Chapter 10, "Commerce Server," focus on the integration of the front–end Web presence and SQL Server data.

Enterprise Application Integration

The purpose of EAI is to integrate applications within an organization. As the need for data–driven applications has grown over the years, so has the number of applications within organizations. Each application is built with a specific purpose in mind. Often a user would later like to perform analysis on data from multiple applications that are storing data in different locations. EAI facilitates the integration of these dissimilar applications within an organization. This section provides an example of an EAI scenario. More important, it focuses on SQL Server's role in the model and the other .NET Enterprise Servers that are being integrated to make the solution happen.

In the example shown in Figure 8.5 you want to integrate two applications. You have a credit card approval system that has been written as a Web application. It currently is using IIS and is storing its data in a SQL Server 2000 database. You recently upgraded the package to perform payroll functions and purchase order approvals, and you eventually would like to integrate several other features that are currently part of the human resources package (401k, benefits, expenses, and so forth). Another department within the organization has created a similar system for credit card approval. That application has been written in Exchange Server 2000 and Outlook 2000 and is using the collaboration and message–routing features of Exchange Server. This department has no intention of enhancing the application to support the additional features.
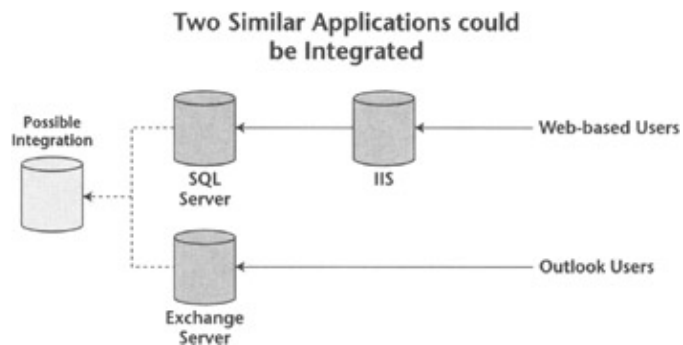
Figure 8.5: It may be desirable to integrate two similar systems to allow seamless user interaction with applications.

You have several key employees who like the interface and functionality of the Exchange Server solution, but you want them to be able to use the Web–based solution for the other functions of the application. You want the applications to integrate so that the users can enter credit card approval information from either application and still use the other features provided by the Web–based application.

As was mentioned in Chapter 2, ".NET Business Solutions," you can integrate applications at one of three layers. In this case it probably would make the most sense to integrate the applications at the data source layer.

You could use a tool such as SQL Server DTS to transfer data from one of the databases to the other. DTS uses OLE DB providers to connect to the data source and destination and transfer the data. DTS also provides the option of transformation. In this example, although the systems are similar in purpose, the database schematics most likely have some differences. You would have to build a transform in DTS to map the data from the Exchange Server system to match the data in the SQL Server system. For example, the credit card information in the Exchange Server system may include fields for credit card type, bank, number, and expiration date. The same information may be stored in the SQL Server system as credit card type, issuer, credit card number, and expiration date. In this example issuer and bank are two separate terms used to describe the same field. You will have to map the two as being the same column. This example is oversimplified, but the potential difficulties are evident. In the integration example this option for integration probably makes the most sense because you have developed both applications internally. You should have access to the database schematics, and so mapping the data is probably easier than it would be if you had purchased two totally different systems developed by different vendors.

In this example integration at the business logic layer is probably not the best approach. Both systems already have been created with their own object models. The components in one application probably are not in use by the other application. In the design phase of the development of both applications it might make sense to create components that could be implemented by both applications and integrate the business logic. At this point, however, rewriting the applications to integrate the business logic probably would be an extensive task.

Business–to–Business Model

In a business–to–business (B2B) model you are attempting to integrate applications and data across organizations. These are applications that assist separate organizations in integrating business processes. In many cases it is advantageous to perform the data integration over the Internet.

This section provides an example of a B2B solution that focuses on the role SQL Server plays in the integration between two organizations. More information on the types of applications that can be

candidates for a B2B solution appear in Chapter 2, ".NET Business Solutions."

In this example, as shown in Figure 8.6, two organizations are tightly integrated. The first organization, Softouch, is an information technology (IT) sales and support company. This organization supports small to medium–size companies as well as one large organization, Organization Big. In an attempt to decrease the number of incoming calls, Softouch created a corporate Web site to which its clients can connect, request customer service, and purchase new products. The Web site was created using Active Server Pages (ASP) on IIS, and the data are stored in SQL Server.
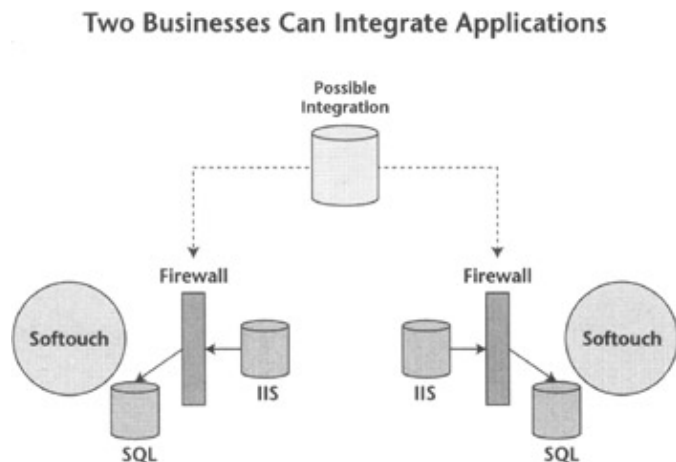
Figure 8.6: Softouch and Organization Big are tightly integrated organizations that would like to integrate their business processes for IT sales and support.

Organization Big currently has an internal application that employees use to request IT products and services. Those requests are forwarded to a primary contact who manually emails the request to Softouch. The application is also a Web–based application that uses IIS and SQL Server.

The two organizations would like to integrate the processes. Organization Big would like to take the primary contact out of the mix and have the IT product and service requests directed straight to Softouch.

The solution can be implemented through BizTalk Server. When the data are exposed as XML, the applications can use BizTalk Server to transfer and map the data from Organization Big to Softouch. More information on the messaging services, orchestration services, and mapping options available in BizTalk Server appears in Chapter 12, "BizTalk Server." In this solution SQL Server 2000 is nothing more than a database that is used at both organizations. It makes the process seamless because it exposes data as XML even though it is just the repository.

Review Questions

1. What is the difference between the Enterprise Edition and the Developer Edition of SQL Server?
2. Why might you want to install multiple instances of SQL Server on a single machine?
3. What is the purpose of federated database servers?
4. What enhancement is provided with XML support for SQL Server 2000?
5. In what types of enterprise solutions does SQL Server play a significant role?
6. What is SQLXML?

Chapter 9: Application Center 2000

Overview

As use of the Internet has increased, the need for applications that are deployed across it has grown as well. As a result of this growth, the applications generally use a Web server to coordinate application communication. The stability and speed of an application are dependent on the Web server with which the client interacts. Application Center 2000 addresses these issues and provides a stable platform on which to coordinate the communication of all applications.

The functionality of Application Center is not limited to Web applications. Internal applications such as intranet and client/server applications can take advantage of these features as well. For customers who are building and managing Web applications, three key issues must be addressed:

- **Manageability.** How easy is it to update and upgrade a Web site's content?
- **Scalability .** How does a Web site cope with additional demand?
- **Availability.** Will a Web site be available 99.99 percent of the time?

Microsoft Application Center 2000 (Application Center) is a management and deployment tool that answers these questions for Web and COM+ applications. Application Center allows developers to take their Web applications to the Internet without the availability and scalability concerns related to most Web applications. Through load–balancing and synchronization services, Application Center helps ensure that a Web application is available, fast, and easy to manage.

This chapter introduces Application Center. Application Center does not have different editions; it does, however, have multiple components. The first section of the chapter describes the different application components and the requirements for their associated installation. The second part focuses on the features that enhance the .NET services. This core part of the chapter primarily introduces the reader to the load–balancing and synchronization services of Application Center. Finally, the chapter addresses the interaction with other .NET Enterprise Servers to create .NET business solutions.

Application Center Installation Components

Application Center 2000 is a little different from many of the other .NET Enterprise Servers. It does not come with different editions and different features that are based on the edition of the product. Application Center has two different application components that the user can install:

- The full server installation, which includes all the functionality that is required for Application Center. The full installation includes everything needed to load balance and synchronize applications.
- The administrative client installation, which allows users to administer clusters remotely by using the Application Center snap–in. This can be installed on any machine that runs Windows 2000 or XP. Administrative clients install all the tools necessary to manage Application Center.

The administrative client is an easy install performed at the machine from which one wants to administer. It is not covered in detail in this chapter. For more information on installing the administrative client, refer to the Application Center product documentation. Application Center also includes a Web administrative client.

For security reasons, the Web–based administrative client supports only a subset of the functionality provided in the Application Center snap–in. Specifically, users *cannot* perform the following tasks:

- Add and remove members
- Set members online and offline
- Synchronize members
- Deploy content
- Modify cluster and member properties

To open the administrative Web client, perform the following steps:

1. Open Internet Explorer.
2. In the Address box, enter http://yourclustername:4242/.

The default port for the Web client is 4242. The port can be changed to another number through the following procedure:

1. Open the Application Center console tool from the Administrative Tools program group.
2. Click to expand Internet Information Services.
3. Expand the cluster on which the port for the Application Center administrative site is configured.
4. Under the expanded cluster, right–click Application Center 2000 Administrative Site and then, on the pop–up menu, click Properties.
5. In the Application Center 2000 Administrative Site Properties dialog box, click the Web Site tab.
6. In the Web Site Identification box (see Figure 9.1), change the value for TCP Port to the port to use for Application Center administration and then click OK.
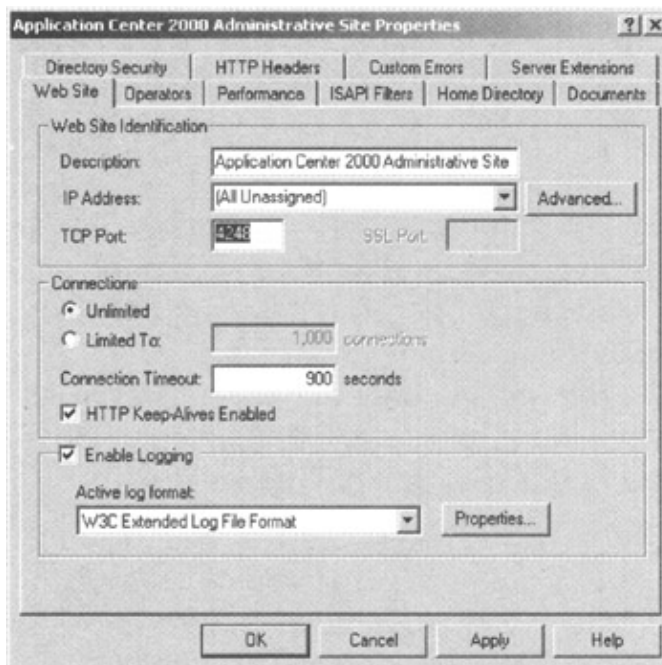
Figure 9.1: The Application Center 2000 Administrative Site Properties dialog box contains the configuration properties related to Application Center. The user can change the TCP port to a number other than 4242 to increase Internet security.

The following section outlines the considerations related to the Application Center Server installation.

Server Installation

The full server installation provides all the features and options of Application Center. It can be installed on .NET Server, Advanced Server, or Datacenter Server. The default installation of Application Center on Windows 2000 Server automatically installs Network Load Balancing (NLB), but NLB is not enabled until the user creates an NLB cluster. Because .NET Advanced Server and .NET Datacenter Server already contain NLB, when Application Center is installed on either of those operating systems, Application Center Setup does not install NLB.

Note All servers added to a cluster must have the same directory and drive structure as the cluster controller. Ensure that the server being added has the same system root, program files path, and Application Center installation directory path as the controller.

If NLB is used with Application Center, the controller and members require at least two network adapters. Additionally, the controller requires one static Internet Protocol (IP) address. This will be the primary (virtual) address of the cluster. The members do not require static IP addresses, although it may be desirable to configure them with static addresses to maintain consistent documentation.

Application Center provides an easy–to–use interface that provides the following advantages for applications:

- **Manageability.** Application Center provides a single image of the content that is installed across the cluster. Your load–balanced servers must have the same content. The synchronization services of Application Center help ensure that all machines have identical content. In Application Center, the application is the content that a cluster serves to its clients. An Application Center application includes all the necessary elements (Web sites, COM+ components, configuration settings, and so on) for a .NET business solution. The application is the content that is deployed across a cluster. Once an application has been defined, Application Center can keep the contents for the application synchronized across all servers that are members of the cluster. The synchronization services can be used to deploy a new version of the application within a cluster to other clusters that need the same application.
- **Scalability.** Client demand on Web sites can change frequently. Application Center can accommodate changing throughput needs because it simplifies the process for adding members to or removing members from a cluster. This is done with no interruption to Web site availability. Furthermore, Application Center uses load balancing to distribute workload throughout a cluster. Application Center is compatible with the leading hardware load–balancing devices. It includes two software load–balancing technologies: NLB and Component Load Balancing (CLB). NLB balances IP requests across a cluster, while CLB balances the activation of Distributed Component Object Model (DCOM) requests. You have a choice as to the type of load balancing to implement. Regardless of that choice, Application Center can scale to a large number of Web requests without decreasing performance significantly as the number of connections increases.
- **Availability.** To provide high availability, Application Center clusters have no single point of failure. If one member fails, your application clients can continue to access the application through other members of the cluster. Application Center has built–in monitoring tools that detect hardware and software failures. When these failures occur, the user can automatically trigger actions such as running scripts and sending email notification in response to the

failures. Additionally, Application Center monitoring is tightly integrated with NLB so that it is possible to configure a member to go offline when an application–level failure occurs. To assist in monitoring system performance, Application Center rolls up event and performance logs across a cluster, enabling a clusterwide view of performance trends and simplifying event management.

Application Center .NET Services

Application Center 2000 runs on top of Internet Information Server. Therefore, it is the primary interaction with the Web user. Microsoft Application Center 2000 provides a number of features designed to simplify the management of Web servers and applications. As has been discussed throughout this book, .NET business solutions focus on making a typical application easier to use throughout an organization.

In many cases this includes porting an application to the Web. When the application is in the Web medium and users are accessing the application from a browser, there is a forum to manage the application easily. Application Center enhances the services of a Web server to help support these applications. This section describes the services Application Center provides to enhance .NET business solutions. The section starts with an explanation of the cluster, which is the primary object in Application Center. Then it describes the types of clusters that can be implemented. This is followed by a description of the load–balancing services of Application Center. Finally, the section details the synchronization services and their role in enterprise application development.

Cluster Services

The basis of Application Center is the cluster, which is a set of servers that serve the same content to cluster users. The cluster user is the application user.

An Application Center cluster can serve internal corporate intranet clients or external Internet clients. The client software can be "thin–client" (such as Web browsers) or "thick–client" (such as Visual Basic .NET) applications. Application Center clusters are designed to manage both the Web tier, which serves HTTP clients, and the business–logic tier, which serves DCOM traffic.

Note Application Center clusters are not the same as clusters defined by .NET Server Clustering. The .NET Server Clustering model is designed to handle database–type traffic such as Exchange Server and SQL Server. .NET Server Clustering uses a shared storage device to coordinate cluster members. Application Center clusters are designed for stateless middle–tier applications such as Web sites and COM+ applications. They do not require a shared disk (or any special hardware). More information on the .NET Server clustering model appears in Chapter 18, "Clustering .NET Servers."

Application Center supports the following cluster types. The sections that follow describe each type of cluster in more depth:

- General (Web–tier) cluster
- COM+ application cluster
- COM+ routing cluster

When you run the New Cluster Wizard, you can choose the type of cluster to create, as shown in Figure 9.2. You must install Application Center on each member in the cluster. When you add a member to a cluster, the following settings and configuration are overwritten on the new member

machine by the settings that are configured on the cluster controller.
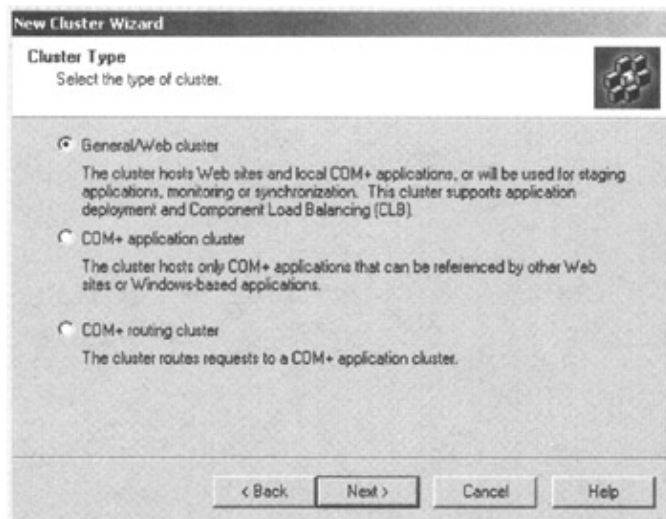
Figure 9.2: The New Cluster Wizard generates a cluster. You are asked to specify the type of cluster you want to create.

To add a server to the cluster, run the wizard to join the server to the cluster. The following procedure describes the steps needed to add a server to an existing cluster:

1. Open the Application Center console tool from the Administrative Tools program group.
2. Right–click Application Center and choose Connect. You are prompted with the Connect to Server dialog box, as shown in Figure 9.3.
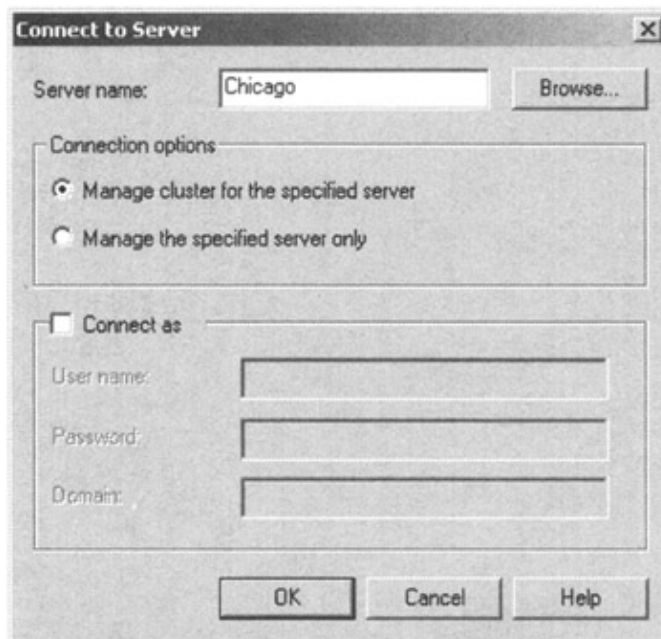
Figure 9.3: In creating or joining a cluster, you provide the security credentials to use when connecting to the server that is running Application Center 2000.

3. Enter the server you want to connect to in the Server Name box. If you are not logged on with administrative credentials, you should use the Connect As boxes to supply the necessary administrative credentials. When you are done, click OK.
4. From the New Server dialog box, select Join an Existing Cluster and then click OK.
5. From the Add Cluster Member dialog box, supply user credentials that have administrative privileges for the cluster controller and the server you are adding to the cluster; then click

Next.

6. Enter the name of the server that is the controller for the cluster, as shown in Figure 9.4, and then click Next.
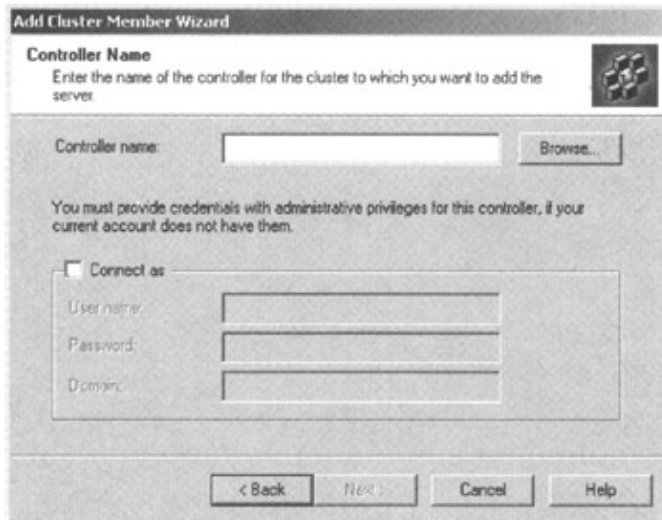
Figure 9.4: The controller for the cluster is the server responsible for managing the synchronization of the cluster.

7. Click Finish to join the server to the cluster.

General (Web−Tier) Cluster

A general cluster is any standard cluster that uses a group of servers to process client requests. These clusters can include the following types of applications:

- Web servers
- Database servers
- Email servers
- Staging/deployment servers
- COM+ load−balanced routing servers
- Standalone servers (clusters with only one member)

COM+ Application Cluster

A COM+ application cluster handles the processing of COM+ components exclusively. When Web servers or clusters instantiate objects for COM+ components, the method calls that instantiate the objects can be load balanced across multiple cluster members within a COM+ cluster. For clusters that handle both COM+ activations and Web sites, the general cluster type should be used.

COM+ Routing Cluster

A COM+ routing cluster uses CLB to route requests for COM+ components from a general cluster to a COM+ application cluster.

> **Note**   Most of the time COM+ routing clusters are not needed. Web clusters can communicate to back−end COM+ clusters over load−balanced DCOM without an intervening COM+ routing cluster. This works because each general cluster member acts as its own router in determining which remote COM+ server should receive remote COM+ component activation.

COM+ routing clusters are useful when COM+ load balancing is required from one general cluster across multiple COM+ application clusters because this release of Application Center CLB is limited to a single list of remote servers for every routing server (that is, every member in a general cluster). This means that if there is a need for load–balanced COM+ activations from one general cluster across multiple COM+ clusters, a COM+ routing cluster is required for load balancing the second and subsequent COM+ clusters. A COM+ routing cluster typically consists of two members that are using NLB to ensure availability of the COM+ router.

Load Balancing

Load balancing distributes workload among cluster members. This is useful for scaling an application. It allows an application to grow and several servers to participate in the processing required to service client requests. Application Center supports three types of load balancing:

- **Network load balancing.** NLB is the IP load–balancing technology that is provided as part of .NET Advanced Server and .NET Datacenter Server. It also ships with Application Center to allow .NET Server to support NLB while running Application Center. With NLB, incoming Transmission Control Protocol (TCP) traffic, User Datagram Protocol (UDP) traffic, and Generic Routing Encapsulation (GRE) traffic requests are distributed among the members of the cluster.
- **Component load balancing.** COM+ allows developers to write software components in a variety of languages. Because the location of COM+ components is transparent, the components can run in a distributed environment. Using CLB, Application Center takes advantage of this to provide load–balanced activation of COM+ components across a cluster. CLB ships as part of Application Center.
- **Other load–balancing devices.** Application Center is compatible with other load–balancing devices. They can be used together, with the deployment and monitoring capabilities provided by Application Center. However, in this version Application Center does not manage these devices actively. The Microsoft Application Center Resource Kit includes software that allows limited integration of other load–balancing devices. This software allows users to set members offline and online from both the Application Center snap–in and the command line. One also can use a Windows Management Interface (WMI) provider (included in the resource kit) to have device monitor the state of cluster members.

Synchronization

Application Center manages the synchronization processes for a cluster. Application Center uses a single application image to represent all the required content for a solution (for example, a Web site). The application image can include Web sites, Web pages, COM+ applications, system settings, and any other content that must be installed on each cluster member.

A cluster controller maintains the latest, most accurate content for a cluster. The controller is responsible for synchronizing its content to all the cluster members. If the content on the controller changes, for example, as a result of an incoming deployment, changes are synchronized automatically to each cluster member. Synchronization also can be performed manually.

Deployment is the transfer from one cluster (usually a stager) to another cluster (usually the production cluster) of content and configuration settings. *Synchronization* is the transfer of content and configuration settings from the cluster controller to one or more cluster members. Other than this distinction, the processes and resources that are used are identical to those used for synchronization and the difference is semantic. Application Center can be used to synchronize or deploy the following settings:

- Web sites, virtual directories [and their associated Internet Server Application Programming Interface (ISAPI) filters], and Active Server Pages (ASP.NET) applications
- COM+ applications
- File system directories and files
- Exportable certificates
- Certificate trust lists (CTLs)
- Data source names (DSNs)
- .NET WMI monitoring settings (in the Application Center namespace, Microsoft Health Monitor 2.1 settings, and event filters)
- Registry keys
- Internet Information Server (IIS) Metabase settings

Application Center uses single–controller synchronization, in which only one cluster member can be designated as the controller at one time and the controller is the authoritative source of all synchronized content and settings. Additionally, Application Center is based on shared–nothing clustering, in which each cluster member has a full set of the content and settings and can function on its own; that is, there are no shared resources such as a redundant array of inexpensive disks (RAID).

Synchronization can occur automatically or can be forced and managed manually. The following section outlines the different options for synchronization.

Synchronization Updates

Application Center provides two types of automatic synchronization as well as manual synchronization. The following automatic updates can be used to keep content and configuration settings consistent across a cluster without the need to intervene.

- **Change–based synchronization.** Application Center receives notification if synchronized content (except for COM+ applications, access control list [ACL] changes, and network settings) on the cluster controller changes. The files that are changed are updated automatically and replicated immediately on all cluster members in the synchronization loop. The user can disable change–based synchronization in the cluster Properties dialog box by clearing the Synchronize Members When Content Is Updated checkbox.
- **Interval–based synchronization.** Periodically, Application Center synchronizes all Application Center applications to ensure that the content and configuration settings are identical across the cluster. The user can set the interval between full synchronizations in the cluster Properties dialog box. The default synchronization interval is 60 minutes. The developer can disable interval–based synchronization in the cluster Properties dialog box by clearing the Synchronize Members When Content Is Updated checkbox.
- **On–demand synchronization.** Application Center provides three types of manual synchronization. The following types of on–demand synchronization can be used to control when and how synchronization takes place within a cluster:
- **Cluster synchronization.** All cluster members in the synchronization loop are synchronized with the controller. All applications are synchronized (except for COM+ applications).
- **Member synchronization.** Only the specified cluster member is synchronized with the controller. All applications except COM+ applications are synchronized. This type of synchronization occurs whether the member is in the synchronization loop or not.
- **Application synchronization.** The specified application is synchronized with the cluster controller on every member in the synchronization loop.

.NET Business Solutions

Application Center 2000 plays a key role in .NET business solutions. It extends the functionality of a user's normal application server to help manage applications. Application Center is not a database server; it is the server with which the users interact. Often the application the user interacts with will have to talk to a database server to get the content of the application.

This section outlines the role of Application Center in .NET business solutions. It begins by outlining the process of deploying an application to Application Center. The section then describes the interaction of Application Center with the other .NET Enterprise Servers. You are provided with examples of how Application Center can enhance enterprise application integration (EAI) and business−to−consumer (B2C) enterprise applications.

Deploying an Application

In the context of Microsoft Application Center 2000, an application is a manifest that lists .NET resources that are synchronized within or deployed to a cluster. An application contains resources by listing references to them; thus, more than one application can list the same resource. When you synchronize or deploy an Application Center application, the resources listed in the application are synchronized or deployed. The resources that can be added to an application include the following items:

- Web sites and virtual directories, which include files, certificates, CTLs, and ISAPI filters referenced by the site or virtual directory
- COM+ applications
- File system directories and files
- System DSNs
- Registry keys

> **Note** Application Center cannot be used to deploy new applications to workstations throughout a network. If you want that functionality, consider Systems Management Server (SMS) or application deployment through Group Policy and Active Directory of .NET Server.

In deploying content to Application Center, one should do so only to the controller. Otherwise the deployed content is overwritten on the next scheduled synchronization. Once you deploy content to a member, you can set this member online and promote it to cluster controller. In this manner all members receive the deployed content when the members are synchronized with the controller.

Because deploying COM+ applications and global ISAPI filters requires the Web service to restart, Application Center does not synchronize these files automatically. To synchronize these files across a cluster, create deployments and include those files in the deployment. You can create deployments by using the New Deployment Wizard. You can run the New Deployment Wizard by performing the following procedure:

1. Open the Application Center console tool from the Administrative Tools program group.
2. Click to expand Application Center.
3. Right−click the cluster and select New Deployment.
4. When the New Deployment Wizard displays, click Next.
5. You are asked for your Deployment Target Options, as shown in Figure 9.5. Give the deployment a name and choose whether the deployment should be distributed inside or outside the cluster; then click Next.
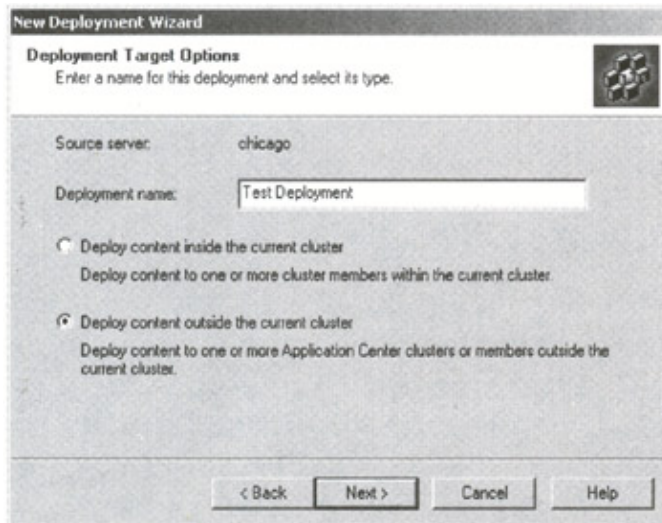
Figure 9.5: The New Deployment Wizard determines the location of the pages you deploy. You can have the content deployed inside or outside the cluster.

6. Enter the user credentials used for the deployment and then click Next.
7. Enter the target content and then click Next to finish the wizard.

When you deploy your own applications, remember that Application Center is not taking care of all your application dependencies. It also is not concerned about the effect on the current users and services. You should consider the following items when deploying your applications:

- COM+ applications may require other resources, such as Active Server Pages (ASP) applications, to work correctly. In addition to deploying an application, you must deploy all the resources that are required by the application.
- COM+ application deployment requires restarting the Web service on the target.
- Application Center synchronizes all content and configuration settings (including COM+ applications and global ISAPI filters) to members when they are added to a cluster.
- COM+ applications are not synchronized automatically. You must deploy them as part of an Application Center application. You can use the New Deployment Wizard to deploy COM+ applications to the entire cluster at the same time, to one or more members at the same time, or to another cluster.
- Application Center restarts the CLB service after COM+ applications are deployed to CLB cluster members.
- Application Center does *not* automatically synchronize file or directory access control list (ACL) changes. Therefore, if you change only the ACL on a file or directory defined in an application, these changes are not synchronized automatically to a cluster. To have these changes picked up during synchronization, change one of the following: the file attributes, LastModified time, or file size (content changes).

Integration with .NET Enterprise Servers

Application Center is a medium between the application user and the data of the application. Thus, Application Center often provides an interface to the .NET database products (SQL Server and Exchange Server). Application Center also can be installed on top of IIS and Commerce Server to add stability and scalability to their applications. Application Center is flexible and can be used to enhance any of your applications regardless of the .NET Enterprise Server you are using. The following sections describe the role of Application Center in EAI and B2C applications.

Enterprise Application Integration

Many organizations struggle with the duplication of applications. As an organization becomes more decentralized, in many cases individual departments start to develop their own applications to track a common business process.

For example, most departments need a purchase order tracking system. One department may have developed a Web–based interface for this application. The application is deployed using ASP on IIS and SQL Server as the database. Another department chose to implement an application for the same type of process by using the routing features of Exchange Server. That application uses a combination of Outlook and Exchange Server. Another department manages the whole process by hand and on paper.

Application Center provides a way to develop the application and make it available all the time. You can use Application Center to deploy your application to the corporate intranet, giving all departments access to the application. You still have to deal with the politics of getting each department to use the application, but Application Center increases application performance and availability, and this should make it an easier sale.

Business–to–Consumer Applications

The most common use for Application Center is for the deployment of applications to customers. Many organizations have a Web presence that presents products to their customers. Because you are not certain about the traffic your Web application will get, your application has some unique challenges. Most Web applications to Web customers have the following requirements:

- The site must be available at all hours.
- The site must be fast regardless of the number of users.
- You need to be able to take a Web server offline for maintenance without taking the site down.
- If the site has multiple Web servers, you want the same content on each server.

Application Center provides all these options for a Web application. You can use Application Center cluster, load–balancing, and synchronization services to provide an answer for each of these B2C application requirements.

For example, you have a Web application that sells books for children. You need to make the site available at all hours because you cater to families around the world. You also want to have the site available at all times to ensure that you do not lose sales to your major competition.

A solution, as shown in Figure 9.6, is to deploy your Web application on Web servers running Application Center 2000. You can deploy the application as ASPs pages on multiple Web servers outside your firewall. The ASPs then connect to an SQL Server behind the firewall to store purchase and product information. The connection to the SQL Server is made over port 1433, which has to be opened on your firewall (Internet Security and Acceleration [ISA] Server). The port only has to be opened from the Web servers to the SQL Server as a tunnel. For more information on ports and tunnels, refer to Chapter 14, "Internet Security and Acceleration Server."
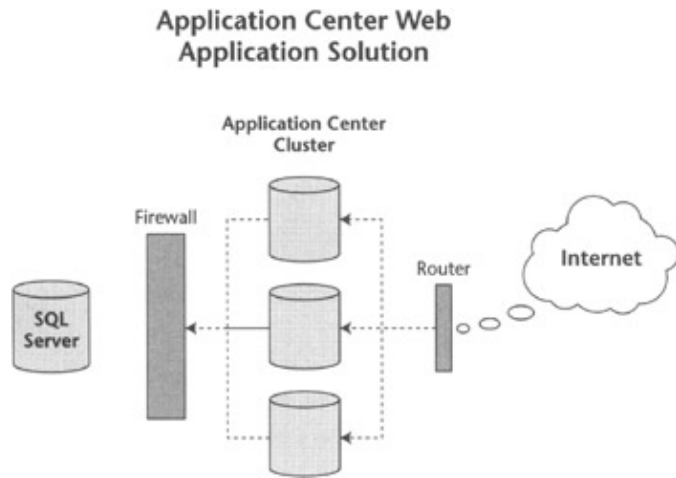
Figure 9.6: A Web application can use Application Center to ensure performance, reliability, and flexibility.

This solution allows you to use the load–balancing services of Application Center to make sure your Web application performs well regardless of the number of users. You can use the synchronization services of Application Center to ensure that all the Web servers have the same content all the time. Most important, you can use the cluster to make certain that your customers can connect to the site even if one of the servers is down. The cluster also allows you to take one of the servers offline for maintenance without affecting the connection to your Web application.

Review Questions

1. What are the reasons you would consider using Application Center 2000?
2. What is the definition of a cluster?
3. What types of load balancing does Application Center 2000 support?
4. What is the role of the cluster controller?
5. What options do you have for configuring automatic synchronization?
6. What is the role of Application Center in EAI solutions?
7. What is the role of Application Center in B2C applications?
8. What is the relationship between IIS and Application Center 2000?

Chapter 10: Commerce Server

Overview

Microsoft Commerce Server 2002 provides a rich and extensible foundation for building commerce applications that can be integrated with back–end databases. Because the architecture is based on the Component Object Model (COM), it is easy to extend the platform by choosing from the large number of integrated third–party solutions available or by having site developers customize Commerce Server tools to meet an organization's business needs. This process has been enhanced by the release of Commerce Server 2002, which takes advantage of several of the .NET services.

Commerce Server 2002 includes the Commerce Server .NET Application Framework, which is an extension of the ASP.NET programming model. Developers can use this framework to create Commerce Server applications that use ASP.NET code to run Commerce Server services and systems.

Commerce Server 20002 is integrated with XML. Several of its features, including Business Desk and Profiles, are based on XML structures. The dependence of Commerce Server on .NET services allows seamless integration with the other .NET Enterprise Servers. The integration also facilitates quicker development because development with Commerce Server is similar to or based on technologies with which many of the developers are already familiar.

This chapter describes Commerce Server. It is a precursor to Chapter 16, "Building Enterprise Web Farms." The chapter begins by outlining the different editions of Commerce Server. This section discusses the dependency on SQL Server. The products are highly integrated. Commerce Server stores all its configuration and customer content in SQL Server databases. The chapter then focuses on the interaction with the .NET services. This part of the chapter focuses on the use of Object Linking and Embedding Database (OLE DB), ASP.NET, and XML.

The chapter concludes with a section on .NET Business Solutions. Commerce Server often provides the authentication and security interface for business–to–customer applications. This final section describes the security options that are enhanced by using Commerce Server 2002. It also explains the role of Commerce Server in business–to–consumer (B2C) applications. Through this process integration with the other .NET Enterprise Server, such as Application Center 2000, is outlined.

Commerce Server Editions

Users of Commerce Server can choose among three editions: the Standard Edition, the Enterprise Edition, and the Developer Edition. Regardless of the edition you choose, you have some basic software requirements to implement Commerce Server. This section introduces the basic requirements of Commerce Server and then outlines the core differences between the editions of the product.

Software Requirements

This section documents which software requirements have to be in place to deploy Commerce Server in an organization. The following sections outline the products that are installed for use with Commerce Server.

Internet Information Server

To deploy a Web site as a commerce site it is necessary to have Internet Information Server. Commerce Server is built on top of the core services of IIS. HTTP and SMTP are used extensively by Commerce Server 2002. The features in IIS that must be enabled at a minimum to implement Commerce Server are:

- **IIS services.** Commerce Server requires the utilization of the HTTP and SMTP services of IIS. The Direct Mailer component of Commerce Server requires access to an SMTP mail server. You may want to have code to send out an order confirmation and may need access to an SMTP mail server. This can be any SMTP mail server, not necessarily the one installed on Commerce Server. Commerce Server does not require the File Transport Protocol (FTP) and Network News Transport (NNTP) services of IIS.
- **HTTP 1.1.** HTTP 1.1 allows for the support of GET, POST, and PUT verbs from a Web browser.
- **File extensions.** The following file types need to be served by IIS: ASP.NET, HTM, GIF, JPG, HTA, HTC, XML, XSL, CSV, CAB, TXT, CSS, CHM, HHC, HHK, JS, VBS, and EXE. JS and VBS are downloaded to the client; they are not run on the server. Commerce Server provides OWC10.exe (Office Web Controls version 10) as a download to the client browser; it is not run on the server.
- **WebDAV.** Web Distributed Authoring and Versioning is used for importing catalog files. The catalog files generally are stored in an XML extension.

SQL Server 2000

Commerce Server is highly dependent on SQL Server. Commerce Server requires SQL Server databases for its connections and data management. The number of databases required by Commerce Server is dependent on the number of Commerce Server features the user chooses to take advantage of. Table 10.1 lists the databases that typically are used with Commerce Server along with their default names. The names and connections can be changed by using Commerce Server Manager.

Table 10.1: Commerce Server Databases

DATABASE	TYPICAL NAME	DESCRIPTION
Commerce Server	*<site name>*_commerce	Stores data related to catalogs, campaigns, transactions, profiles, and events.
Administration	MSCS_Admin	Configuration data used by Commerce Server.
Data Warehouse	*<site name>*_dw	Stores the operational data about users who visit the site. This information is imported from the Web log file and Commerce Server databases by using the Data Transformation Services (DTS) of SQL Server.
Direct Mailer	Direct Mailer	Data for mailing lists, messages, and events.

SQL Server does not have to be installed on the same machine as Commerce Server. In many cases, SQL Server is stored behind the firewall and Commerce Server is outside the firewall. It is important to view Commerce Server from two perspectives. The first is that of the Web user. Commerce Server is used to commerce–enable the site. The second, database perspective is not commonly considered. Commerce Server creates a data warehouse about the users who access a site. This information can be critical in judging the effectiveness of a site. The SQL Server

databases can be accessed either through Commerce Server Manager or through SQL Server reporting tools.

Commerce Server Editions

Commerce Server 2002 is available in three editions. You should evaluate the purpose and features associated with each edition to determine the most cost–effective solution for your organization. The following sections introduce the editions of Commerce Server and highlight the differences between them.

Commerce Server 2002 Standard Edition

The Standard Edition is designed for developing and managing medium–scale Web sites. This edition scales to the following specifications:

- Two processors.
- Two Web servers per application. Up to 10 applications are supported on a single IIS Web server.
- Two Commerce Server applications per site: one Business Desk application and one non–Business Desk application. For example, you might install the Retail Solution Site application or a non–Commerce Server application and then the Business Desk application.
- Ten sites per data warehouse.
- Ten sites per Profiles database.

As these limitations show, the Standard Edition is a solid solution for a small to medium–size organization that is not planning to exceed the number of Commerce applications or sites that are supported by this edition.

Commerce Server 2002 Enterprise Edition

The Enterprise Edition is designed for developing and managing large–scale or high–traffic Web sites. The Enterprise Edition scales to the following specifications:

- 32 processors.
- As many Web servers and Commerce Server applications as a company's hardware supports. Up to 10 applications are supported on a single IIS Web server. These applications can be contained in a single site or distributed across many sites.
- Ten sites per data warehouse.
- Ten sites per Profiles database.

The Enterprise Edition is designed to support sites that require advanced analytics. It provides the ability to analyze user profile data and marketing campaigns and then use the results of that analysis to target the content on a site. The advanced analysis features support the ability to provide real–time recommendations to users visiting the site.

Commerce Server 2002 Developer Edition

The Developer Edition is the commerce solution for developing large–scale, multilingual, highly available sites. This edition has the same features as the Enterprise Edition but cannot be used in a production environment. It has a four–processor limitation. The Developer Edition of Commerce Server 2002 is available with the Developer Edition of Visual Studio .NET. The Developer Edition includes extensive documentation for developers as well as an extended set of samples.

Commerce Server and .NET

Commerce Server 2002 is highly integrated with .NET services. Commerce Server takes advantage of OLE DB, ASP.NET, HTTP, and XML. This section describes the role of .NET services with Commerce Server 2000. To do this effectively, the section begins by outlining a couple of the basic concepts of Commerce Server. This includes a description of sites, Web servers, and resources. This section then discusses interaction with OLE DB. Finally, it describes the role of XML with Commerce Server 2002.

Commerce Server Basics

It is necessary to describe a couple of core terms for Commerce Server. The definition of these terms you will explain the basic configuration of a Commerce site and introduce the reader to some of the interactions between Commerce Server and .NET services.

Site

A site is a collection of ASP.NET–based applications that use base class library (BCL) objects. A site can be created with any language that is compliant with common language runtime. Sites use and modify resources. A site provides a means of administering the applications in that site as a group. This makes it possible to take multiple applications and apply a single configuration to all of them at the site level.

A site generally has two interfaces that are used to present two different perspectives on data. You generally have one application that Web users interact with to view your online store. This is your catalog of products or services. The second interface is from the Business Desk. To view the information about the Web users, the managers of the application use the Business Desk. For instance, you may want to track which users have connected to the site and requested information about a specific product. This interface is used to analyze data about Web users.

A Commerce Server 2002 application is a Web site. It is a logical representation of an application in IIS 5.0, and it appears in both the Commerce Server Manager and the IIS console trees. Commerce Server is highly integrated with IIS and ASP.NET pages. You can view and manage commerce applications from IIS or from Commerce Server. The commerce application represents the Web address that a Web client accesses. When you add an application to Commerce Server, the administration database, which is in SQL Server, is updated with the configuration data for the application. When a Web server is added to the application, the Active Server Pages (ASP) files are installed on the Web server computer. Commerce Server is discussed in the next section.

An application includes its own Web servers, which in turn serve the ASP.NET page content for the Web site. An application always includes at least one Web server and can include multiple Web servers. You can set up a site at the root level of an IIS Web site or at a virtual directory any number of levels below the root.

A Commerce Server site thus is highly integrated with ASP.NET pages, IIS, and SQL Server. One cannot do much with Commerce Server alone. The functionality of the product is defined by its integration with the .NET services and .NET Enterprise Servers. As will be shown later in the chapter, other .NET Enterprise Servers can help create the full enterprise solution. At this point the servers that have been discussed are a prerequisite for any functionality in Commerce Server.

Sites can be managed from the IIS console and the Commerce Server Manager console. The following lists outline the items that can be configured from each of the consoles. IIS has several features that cannot be configured in Commerce Server. Commerce Server Manager has its advantages as well. The primary security extensions can be configured only from Commerce Server Manager.

From the IIS console it is possible to configure the following administrative settings:

- Virtual directory properties
- Default documents
- Directory security
- HTTP headers
- Custom error messages
- HTTPS (Secure Sockets Layer encryption)

From the Commerce Server Manager Console a developer can configure the following settings:

- HTTPS (Secure Sockets Layer encryption)
- IIS application path
- Autocookie
- Authentication filter
- Configuration settings that are common to all the Web servers in an application

More information on Autocookie and the Authentication filter is presented later in this chapter.

To access the configurable properties for a site from the IIS administrative console (Internet Services Manager), perform the following steps:

1. Open Internet Services Manager from the Administrative Tools program group.
2. Double−click to expand the server you want to configure.
3. Right−click the desired Web site and select Properties, as shown in Figure 10.1.
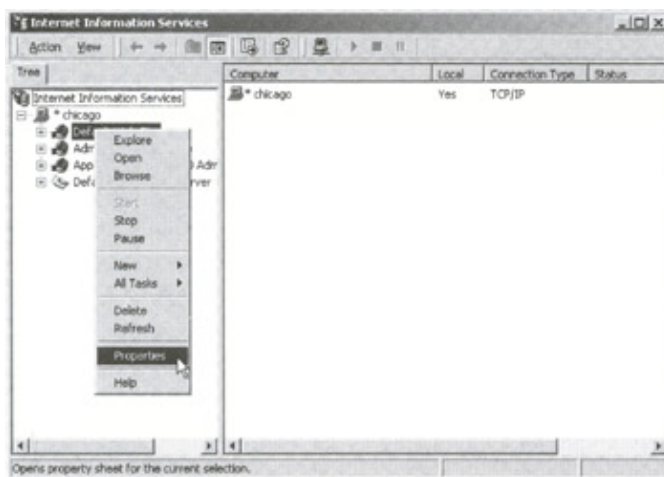
Figure 10.1: Internet Services Manager can configure the settings for a Web site.

4. Select the tabs necessary to configure the desired properties.

To configure settings from the Commerce Server Manager Console, perform the following steps:

1. Open Commerce Server Manager from the Commerce Server program group.
2. Click to expand Commerce Server Manager.

3. Click to expand Commerce Sites.
4. Right–click the application you want to configure and select Properties, as shown in Figure 10.2.
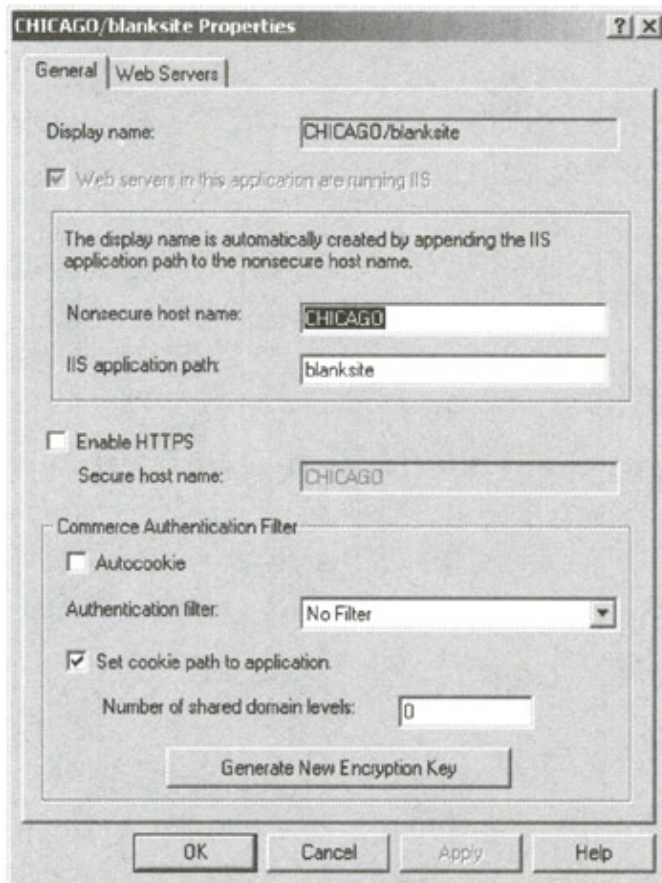
Figure 10.2: Commerce Server Manager changes configuration settings for commerce sites.

5. Select the tabs necessary to configure the desired modifications.

Web Server

In Commerce Server, a Web server is a computer with IIS Version 5.0 or later installed. To use a Web server with Commerce Server 2002, Commerce Server must be installed on the same computer as IIS.

In Commerce Server a Web server is associated with one or more applications; different applications can share the same Web server. The Web server contains the ASP.NET files and subfolders for the application. One application can have several Web servers that, when taken together, form a Web farm.

Resources

A resource is an entity that provides functionality to the applications in a Commerce Server site. A resource consists of one or more COM objects that are used on the Web pages of an application to access the resource functionality. Resources include properties that are stored in the administration database. These properties can be accessed and changed by using Commerce Server Manager.

There are two types of resources: global resources and site resources. Global resources are available for use by all sites. Site resources are available only to the site with which they are

associated.

Commerce Server and OLE DB

The OLE DB provider for Commerce Server serves as the Commerce Server data warehouse manager as well as an SQL data storage engine. Your interaction with Commerce Server for analysis and storage is dependent on the OLE DB provider. As the data warehouse manager, the OLE DB provider for Commerce Server manages the application schema, including the classes, data members, class relations, and surrogate keys. The provider also is responsible for the storage of data warehouse entities. As a typical SQL storage engine, the OLE DB provider for Commerce Server allows the data to be accessed by the application. The provider also optimizes the updates and retrievals from the SQL Server database.

The OLE DB provider for Commerce Server is a COM component that accepts calls to a subset of the OLE DB Application Programming Interface (API) and processes each request against the data source. The OLE DB provider for Commerce Server provides two distinct paths of functionality. The Fastload property that is passed in the connection string determines which of these functionalities is used. When the Fastload property is set to True, user class data can be written only to the OLE DB provider for Commerce Server and cannot be read. When the Fastload property is set to False, data manipulation statements such as Select, Insert, Update, and Delete are supported.

Commerce Server and XML

XML is used extensively by Microsoft Commerce Server 2000. Different features of Commerce Server use XML in different ways. The dependency of the services on XML results in a variety of defined XML structures. These structures define how the data used by Commerce Server are stored and retrieved. Each of the core features of Commerce Server has base XML structures that determine data interaction with the feature.

This section provides an introduction to the types of XML structures Commerce Server uses but does not outline the XML code or the specific requirements of the structures. A reader who is interested in these details can refer to Microsoft's Web site for the XML Reference at http://msdn.microsoft.com/library/default.asp?url=/library/en−us/comsrv2k/htm/cs_rp_xmlrefintro_ejyh.asp. The types of XML structures used by Commerce Server include the following:

- **Analysis/reporting XML structures.** Describe the XML structures used to create reports by Commerce Server analysis. Commerce Server analysis generates reports based on the information in the Commerce Server data warehouse. Typical reports are based on user activity and interaction with the site.
- **Catalog XML structures.** Describe the XML structure in which product catalogs must be represented to be imported into the Product Catalog System. This system stores and retrieves the products that are available through the commerce site.
- **Business Desk XML structures.** Describe the various XML structures used by Commerce Server Business Desk, including those used to define category and module structure and those used by the various HTML components (HTCs). The Business Desk is the primary interface for the administrator or manager of the Web site. This is where the developer can update and view information about the site.
- **Profile XML structures.** Describe the XML schemas and data structures used by the Profile service. Profiles store the information about users who have come to the site. This is helpful to ensure that they can keep the same user information every time they access the site.

.NET Business Solutions Commerce Server can play a large role in your organization for your .NET Business solutions. This should be dependent on the type of organization you have. Commerce Server is particularly important for B2C applications. Commerce Server typically does not play a large role in enterprise application integration (EAI) and business–to–business (B2B) applications. The features provided with Commerce Server are customer–related. The features provided with Commerce Server facilitate the following advantages for your e–commerce site:

- **Authentication.** Commerce Server extends the built–in authentication options of SQL Server and IIS to help customize the user authentication process.
- **Encryption.** You can enable Secure Sockets Layer encryption through Commerce Server. While this is possible with SQL Server and IIS, the process is made easier by Commerce Server.
- **Profiles.** Commerce Server can track user–specific settings for a site to help customize the visitor's experience.
- **Analysis.** The built–in data warehousing features of Commerce Server provide a reporting medium for user data. While it is possible to use SQL Server and analysis services to provide a similar solution, Commerce Server builds in the analysis options and makes them much easier to use.

These features of Commerce Server are tailored for B2C applications, the focus of this section. In most cases organizations focus on the security features of Commerce Server because it provides the interface with the customer. This section outlines the security features of Commerce Server, including both authorization and encryption. The section then provides a short introduction to the strategic placement of Commerce Server in a B2C application.

Commerce Server Security

Security for a commerce site is the top concern of any organization that is putting commerce–related data on the Internet. An organization wants to make sure that the users who access the site are who they say they are and that the data that are transferred over the Web are in a secure and encrypted format. This breaks down the security into two main categories: authentication and encryption. Commerce Server can be used to configure both categories for a site. Commerce Server has built–in technology to deal with the authentication requirements but requires a certificate to implement the required encryption for data transmission successfully. The details on certificates that are briefly outlined in this chapter are described in more detail in Chapter 16, "Building Enterprise Web Farms."

The authentication process is the action in which a user passes credentials (username and password) to the server. The server then is responsible for verifying the validity of the credentials and granting or denying access to the server. Commerce Server provides two objects for developers to use for authentication: AuthManager and AuthFilter. The following sections describe the details of using each of these objects and the authentication options they provide through IIS. This section ends with an overview of the encryption options in Commerce Server.

AuthManager

AuthManager is a COM object that exposes methods for identifying users and controlling access to dynamically generated content. The AuthManager object is installed with Commerce Server 2000 and later versions and provides the foundation for all the authentication methods described in this section.

The AuthManager object identifies users and gathers the information required for user

authentication. This information is stored in an object called a ticket. The AuthManager object uses cookies and encoded URL strings to store the ticket, providing a transparent interface to the user information. It also manages encryption and decryption processes to ensure the security of the ticket. Encoded URL strings can be used to pass ticket information between pages without the use of cookies. This method works only when the AuthManager object generates all URLs. This is the only authentication method provided by Commerce Server that does not require the use of cookies.

The AuthManager object originally was designed to be accessed with ASP.NET scripts; however, a C++ programmer who is coding more complex objects also can access the AuthManager object functionality by using the IMSCSAuthManager interface. For more information on the C++ and Visual Basic .NET (ASP.NET) interface to the AuthManager object, refer to the Programmer's Reference in Commerce Server 2002 Help.

The following authentication methods are offered by IIS and can be used alone or in conjunction with the Internet Server Application Programming Interface (ISAPI) filter AuthFilter provided by Commerce Server:

- Anonymous authentication
- Basic authentication
- Integrated Microsoft Windows authentication
- Certificate authentication

Each method is discussed in detail in the following sections. The authentication method that is suitable for a business depends on the objectives of that business's site.

Anonymous Authentication

Anonymous authentication allows full access without requesting a username or password. This authentication service is the most basic option provided by IIS. Since this method contains no user ID, Commerce Server cannot query it for user information. As the most basic authentication method, it traditionally has been the most commonly used method for Web site authentication. It is also the least powerful option in that it does not support advanced features unless one adds the use of cookies or other intrusive methods. When users connect to the site and you authenticate them as anonymous, the IIS then maps the anonymous users to a valid Windows account. The permission the user has to the data is dependent on the Windows account to which an anonymous user is mapped. The account that is used for the anonymous connections is defined in the configuration of the site from IIS. To view the account that is used for anonymous connections, perform the following steps:

1. Open Internet Service Manager from the Administrative Tools program group.
2. Double−click to expand the server you want to edit.
3. Right−click the site you want to view and select Properties.
4. Select the Directory Security tab, as shown in Figure 10.3.
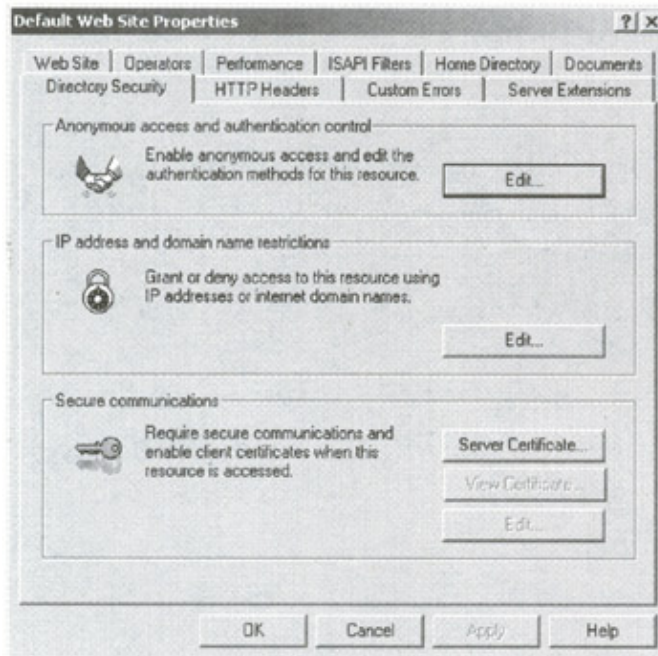
Figure 10.3: The Directory Security tab configures the authentication requirements for a Web site.

5. In the Anonymous access and authentication control, click the Edit button.
6. In the Account used for anonymous access section, click the Edit button to display the current anonymous access account.

The advantages to anonymous authentication include the following:

- **Cookieless browsing.** Anonymous authentication does not require the use of cookies.
- **No logon.** Anonymous access means no delays or impediments for the users. For example, potential users can immediately browse the online catalog without completing a logon or registration process.

The following disadvantages are associated with anonymous authentication:

- **Not a trusted relationship.** There is no way to know or validate a user's identity. There is no way to allow a user access to order and account information without logging in and establishing a secure relationship.
- **Lost user data.** Anonymous validation allows only the most primitive user data to be tracked, and these data can be lost when the user logs on and transitions to a more secure form of validation.

Basic Authentication

Basic authentication prompts the user for a username and password that then are sent unencrypted over the Internet or the company network. Basic authentication is used widely for collecting usernames and passwords despite the lack of security, because it works seamlessly with any type of browser. Basic authentication proceeds as follows:

1. The client computer Web browser displays a dialog box where users enter a previously assigned Windows 2000 account username and password.
2. Using this information, the Web browser attempts to establish a connection to the server. The password is Base64 encoded before it is sent over the network.

143

3. If your server rejects the information, the Web browser repeatedly displays the logon dialog box until the user enters valid information or closes the dialog box. If the Web server verifies that the username and password correspond to a valid Windows 2000 account, a connection is established.

The only advantage of basic authentication is that it supports almost all browsers. The disadvantage is that all passwords are transmitted in an unencrypted format. Any network monitor tool can easily capture usernames and passwords. Because of the unencrypted format, you should consider Basic authentication only in an HTTPS/SSL environment.

Note   HTTPS and Secure Sockets Layer (SSL) take advantage of certificate security and provide an encrypted format of data transfer over insecure mediums such as the Internet.

Integrated Windows Authentication

Integrated Windows authentication is based on Windows authentication, which employs hashing technology to identify users without sending passwords over the Internet. When you enable integrated Windows authentication, the user's browser communicates the password in a cryptographic exchange with your Web server. Authentication credentials are processed in only a one−way method. The result of this process is called a *hash,* or message digest, and is extremely difficult, if not impossible, to decrypt.

Integrated Windows authentication, which formerly was called NTLM or Microsoft Windows NT Challenge/Response authentication, is a secure method of authentication because usernames and passwords are not sent across the Internet, where they can be sniffed or otherwise attacked. Integrated Windows authentication proceeds through the following steps:

1. Unlike basic authentication, Integrated Windows authentication does not initially prompt users for username or password information. Instead, it uses the current Windows user information on the client computer for authentication.
2. If required, Internet Explorer Version 4.0 or later can be configured to initially prompt for user information.
3. If the authentication exchange initially fails to identify the user, the browser will prompt the user for a Windows account username and password, which it then will process by using Integrated Windows authentication.

The advantage of Windows authentication is that it is very secure. The disadvantages include the following:

• Only Microsoft Internet Explorer Versions 2.0 and later support this authentication method.
• Integrated Windows authentication does not work over HTTP proxy connections.
• Additional Transmission Control Protocol (TCP) ports must be opened in the firewall.

Certificate Authentication

Certificate authentication allows a trusted relationship between your site and your users that is based on the use of previously issued digital certificates that establish identities and allow the site to enter into a secure connection with the client. By using the SSL security features of your Web server, you can do the following:

• Use a server certificate to allow users to authenticate your Web site before they transmit personal information such as a credit card number.

• Use client certificates to authenticate users who request information from your Web site.

SSL authenticates by checking the contents of an encrypted digital identification submitted by a Web browser for the user during the logon process. Users obtain client certificates from a mutually trusted third–party organization. A server certificate usually contains information about your company and the organization that issued the certificate. A client certificate usually contains identifying information about the user and the organization that issued the certificate. The advantages to certificate authentication include the following:

• Server certificates are useful when you want to open a secure short–term channel to a client for the exchange of password and credit card information.
• Client certificates are timesaving and reliable when extensive traffic can be expected from a single user, as with a B2B relationship between suppliers and large customers.

The disadvantage of certificate authentication is that the requesting and managing certificates can be complex and relatively expensive to purchase.

AuthFilter

AuthFilter is an ISAPI filter supplied by Commerce Server to augment existing IIS authentication methods. The AuthFilter ISAPI filter is actually a dynamically linked library (DLL) that is loaded into the IIS process when a Web service is started. AuthFilter alters the default behavior of IIS and affects the way in which HTTP requests and responses are handled. When AuthFilter is notified of an incoming request, it automatically checks for a correct URL, cookie support in the browser, and a valid user ID or MSCSAuth ticket. The AuthFilter is enabled by default. To disable the AuthFilter, perform the following steps:

1. Open Commerce Server Manager from the Commerce Server program group, as shown in Figure 10.4.
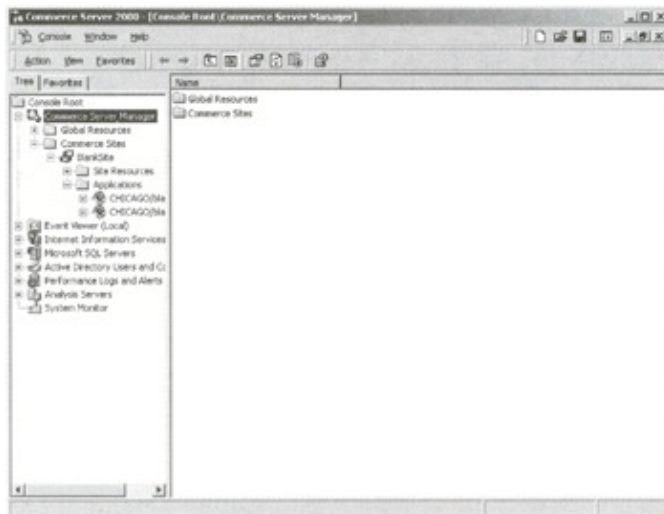
Figure 10.4: Commerce Server Manager is the primary administration tool for Commerce Server.

2. Click to expand Commerce Server Manager.
3. Click to expand Commerce Sites.
4. Click to expand the site you want to edit.
5. Click to expand Applications.
6. Right–click the application you want to configure and then select Properties.

7. In the Properties dialog box in the Authentication Filter option, click the drop–down box to select No Filter, as shown in Figure 10.5.
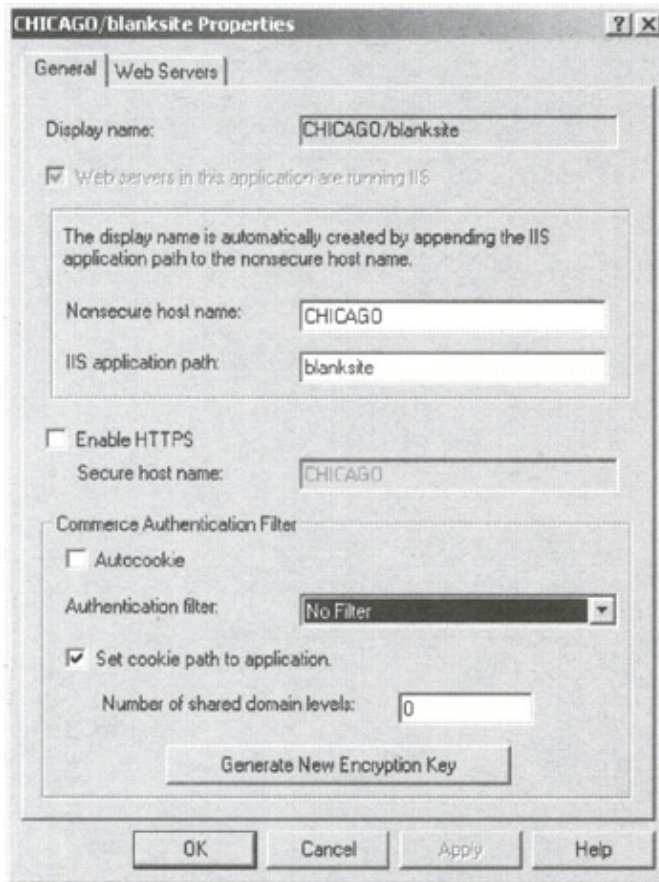
Figure 10.5: By selecting No Filter the user has disabled AuthFilter.

The following types of Authentication are available with AuthFilter. The following sections provide more details on each of these authentication types:

- Windows authentication
- Custom authentication
- Autocookie support

Windows Authentication

AuthFilter can be configured to use Windows authentication to control access to a company's site. In Windows authentication mode, the server validates a user by setting a session cookie that contains an MSCSAuth ticket. No expiration date is specified for the cookie. The cookie is deleted from the user's computer when the session expires.

An MSCSAuth ticket contains a user ID, the last logon time, and a time window specifying how long the ticket is valid after the last logon time. However, a validated user does not automatically have access to a requested URL. After the credentials of the user are validated, they are checked against the access rights of the URL that are maintained through NT file system access control lists (ACLs). For more information about ACLs, see the Windows 2000 Server documentation.

When AuthFilter is employed in combination with Windows authentication mode, AuthFilter looks up the user's permissions in Microsoft Active Directory or a local security access manager account.

When a user logs on to a site by using this authentication method, AuthFilter retrieves the username and password from the HTTP request and stores it in a cache. It also sets a cookie that is valid only for the duration of the user session. Other pages then can use this cookie to identify the user during the session, enabling you to present custom content and targeted offers and track usage statistics.

The advantages of using Windows authentication with AuthFilter are as follows:

- This method provides integration with Active Directory security.
- Windows authentication supports a Web farm scenario (two or more Web servers that are connected to share resources and request loads) with a single logon.
- This method supports the use of proxy accounts.

The disadvantage of this form of Windows authentication is the required use of cookies.

Custom Authentication

This AuthFilter operating mode allows the site developer to employ non−Windows authentication services while continuing to use the base services of AuthFilter. Custom authentication differs from Windows authentication in that it does not provide Active Directory security integration; thus, no security context is provided. It also secures the site at the application root directory and provides no discretionary permissions to subdirectories or pages.

For example, you can use the Profiling System to retrieve site access permissions from the user profile when a user logs on. The Profiling System retrieves user information from a reliable data source. This data source could be a Microsoft SQL Server database, the Windows 2000 Active Directory, a valid Lightweight Directory Access Protocol (LDAP) Version 3 store, or another OLE DB−compliant data source.

In custom authentication mode, AuthFilter performs the following steps after being notified that an authentication event has occurred:

1. Checks for site configuration properties in the local site cache. If the properties are not found, AuthFilter reads the site configuration properties from the administration database by using a SiteConfig object and then stores them in the site cache.
2. Detects whether the requested URL is correct and automatically corrects for case sensitivity in the URL.
3. Checks for session−cookie support and, if it is unavailable, redirects the user to the Active Server Pages (ASP.NET) page specified in the s_NoCookie_Form (No−Cookie form in the Commerce Server Manager user interface) property of the Commerce Server authentication resource. Usually this page notifies the user that cookies are required and that the user should resubmit the request after cookies are enabled.
4. Determines whether the cookie contains a MSCSAuth ticket and, if it does not, redirects the user to a logon page.
5. If the cookie contains an MSCSAuth ticket, checks the current time against the last logon time on the ticket to see if it is within the allowable time specified in the ticket.
6. If the current time is past the time window specified in the ticket, the user is redirected to the logon page as a nonvalidated user.
7. If the current time is within the time window, the ticket is considered valid and the user is allowed to access the requested page as a validated user. If the current time is within 5 minutes of the last logon time plus the time window, the last logon time on the ticket is changed to the current time so that an active user can continue browsing without interruption.

The advantages of custom authentication are as follows:

- Custom authentication allows site designers to customize access to the site by using the base services of AuthFilter without using an Active Directory infrastructure.
- Custom authentication also may be used with Autocookie. When custom authentication is used with Autocookie, it is referred to as mixed–mode authentication.

Custom authentication also has disadvantages:

- Does not provide granular resource authorization for subdirectories or pages.
- Does not provide a security context specific to the user.

Autocookie Support

Autocookie is a service provided by AuthFilter. When the Autocookie mode is enabled, the AuthFilter automatically generates a persistent cookie (MSCSProfile ticket) that treats the user as an anonymous (guest) user. If Autocookie mode is selected, AuthFilter will always check whether the user's browser already has this anonymous cookie. If the anonymous cookie is not found in the browser information, the filter redirects the user to the Autocookie form and begins tracking user information.

Autocookie mode enables Commerce Server to track anonymous user behavior on a site. If a user with an anonymous cookie registers, Commerce Server can update the corresponding persistent cookie and all the profile data gathered while the user is considered anonymous for future analysis. The code must be implemented to update the persistent cookie.

The advantages of Autocookie are that it tracks anonymous users and collects user profile data for future analysis. It also integrates anonymous profiles with the corresponding registered user information when an anonymous user later registers during a session. The disadvantage of Autocookie is its reliance on the use of cookies. The following steps can be performed to disable Autocookie mode:

1. Open Commerce Server Manager from the Commerce Server program group.
2. Click to expand Commerce Server Manager.
3. Click to expand Commerce Sites.
4. Click to expand the site you want to edit.
5. Click to expand Applications.
6. Right–click the application you want to configure and then select Properties.
7. From the General tab, click to clear the Autocookie checkbox.

 Note After you disable AuthFilter or Autocookie, you must restart the IIS services.
Using AuthFilter with AuthManager

When enabled, AuthFilter adds the following features to IIS authentication:

- **Detects whether the requested URL is correct.** AuthFilter automatically corrects for case sensitivity in IIS virtual directory roots.
- **Determines whether a browser supports cookies.** If the client browser does not support cookies, AuthFilter redirects the user to a no–cookie page; AuthFilter requires that browsers support cookies.
- **Determines whether the user has a valid ticket.** Provides an HTML forms logon page instead of the pop–up logon dialog box.

Encryption

Now that the issues related to authentication have been addressed, it is appropriate to point out the options for encryption. Commerce Server takes advantage of the certificate security available in IIS. Through a certificate and the use of public and private key technologies, it is possible to implement SSL encryption. The request from the browser is made with the HTTPS protocol.

To implement encryption successfully it is necessary to acquire and install a certificate on your server. If the Commerce Server will be accessed only as part of your intranet solution, you can distribute and manage your own certificate by taking advantage of the certificate services available with .NET Server. For more information on installing and configuring certificate services, refer to the help files included with .NET Server.

If your Commerce Server needs to be accessible to the general public, you will have to get the certificate from a known certificate authority (CA). The most popular CA in use today is VeriSign. For the steps involved in acquiring and installing a certificate from a known CA, contact the CA to obtain a detailed requirements list. After you have obtained and installed the certificate on your server, you can implement SSL encryption on your site.

To enable encryption from Commerce Server Manager, perform the following steps:

1. Open Commerce Server Manager from the Commerce Server program group.
2. Click to expand Commerce Server Manager.
3. Click to expand Commerce sites.
4. Click to expand Applications.
5. Right–click the application you want to configure and select Properties.
6. From the General tab, click to enable HTTPS, as shown in Figure 10.6.
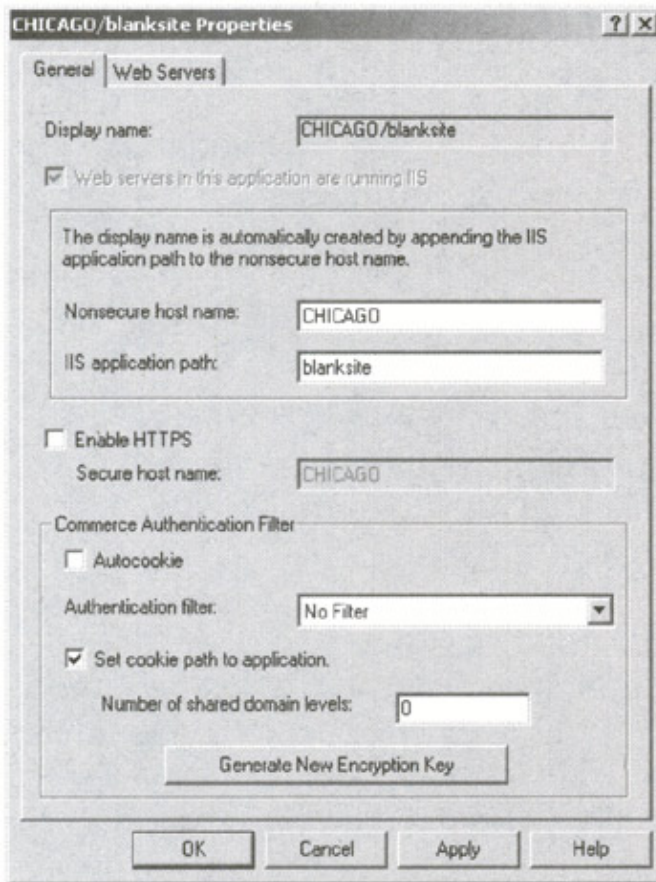
Figure 10.6: Commerce Server Manager can enable HTTPS.

The Role of Commerce Server in Business–to–Consumer Applications

Commerce Server features enhance the user's experience at a company's site. The Web user interacts with the site, and Commerce Server enhances the normal Web connection to IIS. This is accomplished by making the Web site more secure, more customized to the user, and more up to date. Because of this, Commerce Server typically is installed on the machine where your Web site exists. For most organizations this is outside the firewall in the demilitarized zone (DMZ). Alternatively, you could host the site behind your firewall by using the server publishing features included with the Internet Security and Acceleration (ISA) server.

The databases that store product information and user profiles typically are stored on an SQL Server that is behind your firewall. The Web user connects to your site, and the ASP that make up your site connect to the database behind the firewall. You also could extend the availability of your commerce site by integrating the load–balancing services with Application Center 2000.

For example, say you have a commerce site that sells your software products. The product information is stored in the Catalog Product System database on your SQL Server. You need to ensure that the site is available at all times and that you can handle many hits at the same time. A solution, as shown in Figure 10.7, is to integrate Commerce Server with Application Center 2000.
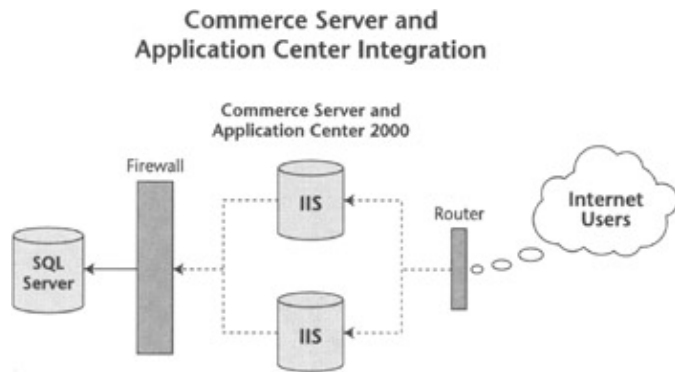
Figure 10.7: Commerce Server and Application Center can be integrated to load balance commerce−enabled sites.

In this example two Web servers are placed outside your firewall. The Web Servers are running IIS and Commerce Server. Application Center 2000 also has been installed. Application Center performs the load−balancing services and synchronizes the ASP.NET application files on both servers. The SQL Server that stores the user profiles and the product information is stored behind the firewall.

This solution is not necessarily a Web farm. In this case the site has been duplicated across both servers. The load−balancing service alternates the connections between the servers. The load balancing makes it possible to scale the application for multiple users. It also helps ensure that the site is always available. With a Web farm you are spanning an application across multiple servers, similar to what has been done in this example. The entire site does not have to be duplicated. Some of the application content could reside on one server, and some of the content on another server. While this example depicts a Web farm, Web farms have many more features and options available to them.

Review Questions

1. What are the steps in a commerce site maintenance life cycle?
2. What are the software prerequisites for implementing Commerce Server?
3. What are the advantages of the Enterprise Edition compared with the Standard Edition of Commerce Server?
4. What are the differences between Windows authentication and custom authentication?
5. What is an Autocookie?
6. Can more than a single Web server be part of a site and/or an application?

Chapter 11: Content Management Server

Overview

Content Management Server 2001 is an enterprise Web content management product that allows an organization to build, deploy, and maintain highly dynamic Web sites quickly and efficiently. Designed for even the largest enterprise, Content Management Server enables a small number of administrators to manage and publish a large amount of content for multiple sites, each with its own purpose.

Content Management Server reduces the time required to build and deploy highly structured Web applications. It includes sophisticated site management and design control tools in a distributed authoring and publishing environment. Content Management Server features reliability and scalability as an extension to the normal Web applications that are deployed with Internet Information Server (IIS). Content Management Server makes the development of Web applications easier for a developer, who can build full–featured Web sites that are easy to maintain. The following features can be applied to Web solution applications:

- **Content contribution for nontechnical business users.** An easy–to–use browser–based tool enables nontechnical users to create and manage their own content.
- **Role–based workflow and approval.** Built–in workflow and approval processes ensure that content is always accurate and up to date on published Web sites.
- **Personalized and targeted content.** Web pages, dynamically rendered in real time, can be classified and personalized by associating meta data properties with content.
- **Content reuse across applications.** The content of a Web site can be stored and managed separately from its design elements, facilitating content reuse and integration with other e–business applications.
- **Rapid time to market.** Industry–standard technology and a complete publishing Application Programming Interface (API) enable rapid development of Web sites.

This chapter discusses Microsoft Content Management Server (MSCMS). It is important to recognize that a great deal of the power of this product lies in its integration with some of the other .NET Enterprise Servers. Content Management Server helps enhance the functionality of IIS, Commerce Server, and SharePoint Portal Server. It also is tightly integrated with SQL Server.

This chapter describes the different applications that are included with Content Management Server and then explains the purpose of Content Management Server. It then outlines the features of Content Management Server for .NET Web application development. It concludes by describing the integration of Content Management Server with the other .NET Enterprise Servers.

Content Management Server Components

Content Management Server is delivered with the following primary components:

- **Content Management Server 2001.** This is the server–side application. It generally is installed on a .NET or Windows 2000 server.
- **Content Management Server Site Builder.** This is a client application that can be used to build and design Web sites.

Content Management Server has several requirements for installation and configuration. The other components are client applications that are deployed on workstations instead of on the server. The following sections outline the additional requirements for each of these components.

Content Management Server

This Enterprise Server has several requirements that must be met for installation. The following list outlines the core requirements for installation of the server piece of Content Management Server:

- .NET or Windows 2000 Server, Advanced Server, or Datacenter Server
- IIS Version 5 or later
- SQL Server Version 7.0 or later
- 500 megabytes (MB) of disk space for Content Management Server install files
- 1 gigabyte (GB) of disk space for the database files
- 150 MB free for deployment and dynamic content processes

After meeting the minimum requirements a developer can install Content Management Server. Once the server product is installed, the database components have to be configured. Content Management Server requires SQL Server and uses a database to store configuration and content information.

The Database Configuration Application (DCA) is available after Content Management Server has been installed to configure the Web sites and database connectivity information. It is necessary to know the following information before running the Database Configuration Application.

- **The Web site that will be the connection point to Content Management Server.** The connection point often is referred to as an entry point.
- **The Web site that will be the entry point for the Server Configuration Application (SCA).** The SCA is used to configure the server–level settings of Content Management Server.
- **The primary NT user account that will be the system account.** This is the system account that is used for the functionality of Content Management Server. The account must have access to the database and have read access to Active Directory.
- **The SQL Server instance to connect to.**
- **The database for the storage of Content Management Server configuration and content.**
- **The account in SQL Server that Content Management Server should use when connecting to the database.**

After you are ready with these items, run the Database Configuration Application. (The SQL Server database you are connecting to should already have been created. For more information on creating an SQL Server database, refer to SQL Server Books Online.) To run the DCA you need to be logged into your machine with an account that is a domain administrator and have database owner rights to the database you will be using. Perform the following procedure to run the DCA:

1. Open the DCA from the Microsoft Content Management Server program group.
2. Click Next to begin the wizard process.
3. Select the Web site that is the entry point for Content Management Server and then click Next, as shown in Figure 11.1.
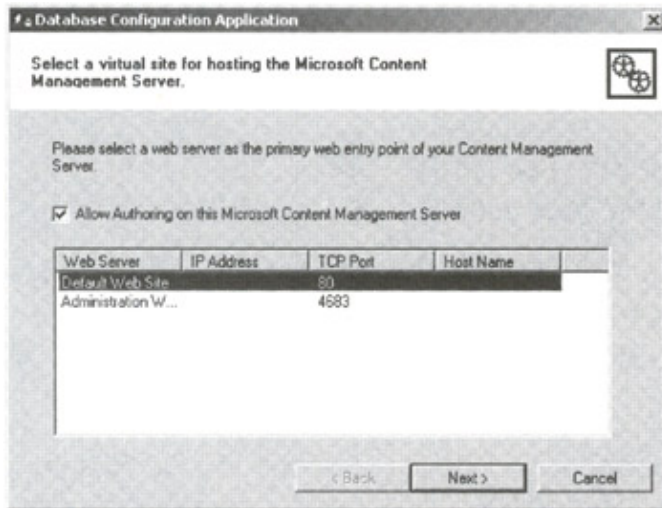
Figure 11.1: The Database Configuration Application configures the Web server that is the primary connection point for Content Management Server.

4. Select the Web site that is the entry point for the SCA and then click Next.
5. Select the primary NT user account for connectivity to the database.
6. Select Yes to allow the IIS services to restart.
7. Click Next to select SQL Server as the database platform.
8. Enter the SQL Server login information as shown in Figure 11.2 and then click OK.
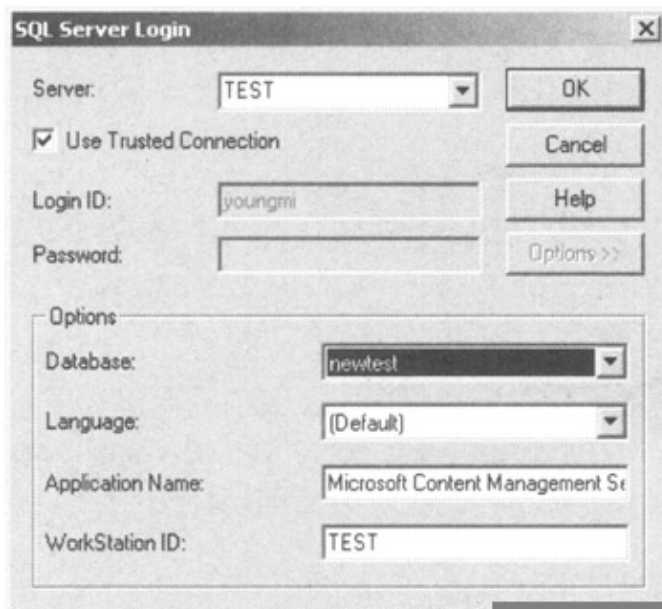
Figure 11.2: Content Management Server connects to the SQL Server instance and database that the user selects.

9. Click Finish to complete the wizard process.

After you have set up the DCA, you can run the SCA. The SCA is the primary configuration tool for Content Management Server. It is a Web–based application that can be used for the following tasks:

- Set security to allow varying levels of viewing access
- Change the background cleanup settings
- Change the location and size of the disk cache
- Change the size of the memory cache
- Set multiple Web entry points

154

- Select or remove Windows 2000 domains
- Change user authentication
- Change the MSCMS 2001 system account
- Change cookie settings

To open the SCA, perform the following steps:

1. Open Internet Explorer.
2. In the Address box, enter http://yourdomainname/nrconfig.

Refer to the online help at Microsoft's homepage for Content Management Server to find more information about the configuration of each of these settings. Additional information can be viewed at www.microsoft.com/cmserver.

Microsoft Content Management Server Site Builder

Site Builder is a client access tool for creating and managing sites that are deployed through Content Management Server. Site Builder has the following requirements for installation:

- Windows 98 or later
- Internet Explorer 5.5 or later
- 128 MB of RAM
- 20 MB of disk space

To install Site Builder, insert the Content Management Server CD−ROM and begin the installation. When prompted, you will have to select the option to install the Site Builder tool, as shown in Figure 11.3.

Figure 11.3: The client Site Builder tool is installed from the Content Management Server CD−ROM.

Purpose of Content Management Server

Content Management Server is a powerful product that can enhance and standardize Web site development. The features of Content Management Server have the following goals:

- Managing Web content
- Delivering dynamic data
- Providing fast deployment of Web sites

Managing Web Content

Several of the features included with Content Management Server help a developer manage comprehensive Web sites. The following list is an overview of the features that enhance the management of Web content. To find implementation details on any of these features, refer to the help files in Content Management Server.

- **Real–time content updates.** This enables content contributors to publish content directly onto development, staging, or live production Web servers.
- **Revision tracking and page archiving.** As pages are updated, existing versions are archived automatically. Users can compare changes of previous work with existing pages.
- **Dynamic server clustering.** Intelligent caching enables clustering of dynamic content servers. This enables load–balanced environments that provide site scalability and server failover.
- **Object caching.** Caching of content objects in RAM and on disk ensures that dynamic page assembly and serving are fast.
- **Content scheduling.** By using automated scheduling tools, users can schedule content publication and archival times.
- **Workflow applications.** By using workflow, users can set up applications that include multiple levels of approval. Workflow is described in more detail in the section titled *.NET Development Features* on page 212.

Delivering Dynamic Data

Dynamic content has become a primary focus of Content Management Server, many of whose features help facilitate dynamic delivery of content. The core features included with Content Management Server for the delivery of dynamic data include the following:

- **Dynamic template switching.** When one uses the publishing API, presentation templates can be switched on the fly, changing the layout or design of the page in real time.
- **Connected content pages.** Allow content to be published through multiple presentation templates to multiple locations or Web sites.
- **Dynamic page assembly.** Content objects and templates are assembled as pages as requested from the Web server.
- **Separation of content from format.** Content is stored and managed separately from its associated presentation templates.
- **Language–specific content targeting.** Allow content contributors to target localized content objects to specific users on the basis of individual language preferences.
- **Multilingual sites.** Allow a single site to contain content for multiple languages. More information on this feature is supplied later in this chapter in the section titled *.NET Development Features* on page 212.

Fast Deployment of Web Sites

One concern with the deployment of Web sites is the time it takes to deploy content to the Web. After changes are made or when a site needs to be redeployed, the time needed to redistribute the site can be detrimental. One of the primary design goals of Content Management Server is the fast deployment of sites. Many of the features included with the product are targeted at that goal. Consider the following features:

- **Quick installation.** Content Management Server provides simple installation wizards.

- **Template and resource galleries.** Templates and Web site resources are managed centrally on the server through template galleries and resource galleries to ensure centralized control over corporate publishing and design standards.
- **XML.** Templates that publish content in Extensible Markup Language (XML) format can be built easily.
- **Site Deployment Manager.** Allows site administrators to move content and Web sites between servers easily.
- **Flexible Component Object Model (COM) API.** Developers can build powerful content management applications and share content with other systems by using the flexible publishing API.
- **Sample templates and Web sites.** Sample templates, Web sites, and customization code are included in the box.

.NET Development Features

Among the features introduced in the preceding section, a few are critical to developers when they are creating .NET solutions. This section outlines some of those key features. The first part of the section introduces the interaction of MSCMS, Active Server Pages (ASP), and ASP.NET technologies. The section then introduces importing and exporting XML data by using Content Management Server. Finally, the section focuses on workflow, which is highly beneficial in designing applications that flow throughout an organization. Workflow is helpful in maintaining Web site standards and creating an approval process.

Interaction with ASP and ASP.NET

Content Management Server interacts with ASP and ASP.NET in several ways. First and foremost is the content that is being published. The content can be ASP or ASP.NET pages. Often, as Web sites grow, they become more difficult to manage. As multiple developers get involved, there may be problems in controlling the content. The content of the site can become distributed, insecure, and chaotic compared with organizational standards. This occurs because the Web site structure grows faster than does the infrastructure that is supporting it. Your developers should be given standards and a process in which to develop. The section titled *Workflow* on page 215 shows how MSCMS provides this infrastructure. The ASP.NET and ASP content must abide by the same rules as the rest of the content.

The second area of interaction between MSCMS and ASP and ASP.NET comes in MSCMS's use of ASP pages to perform some of its work. MSCMS includes a feature called the Web Author. The Web Author is a set of server–side scripts (ASPs) that generate an editable version of an existing or new MSCMS 2001 posting. The Web Author allows authors, editors, and moderators to switch a posting among Live mode, Edit mode, and Authoring mode. With MSCMS 2001, resource managers can add, delete, edit, or replace resources. At the time of this writing the Web Author scripts are ASP pages and do not use ASP.NET. It is expected that in the future the technology will be based on ASP.NET.

The main entry point for retrieving MSCMS publishing API objects (such as postings and channels) is through the Autosession object. Autosession is the object in the publishing API that relies on ASP. After Autosession has been initialized with the parameters of a HTTP request via an ASP page, the other objects can be retrieved and manipulated as required.

Integration with XML

Whether your existing Web site uses XML or XML data files written within another application, you can write a custom script that imports XML information in concert with the MSCMS 2001 publishing API. This API is an object model that the MSCMS server exposes; the publishing API provides the necessary layer of interaction with MSCMS to facilitate the transfer of XML−based information to and from MSCMS.

By using the publishing API in concert with a custom ASP script, you can import existing information formatted by using XML into your Content Management Server database. During this import process the content is removed from its XML formatting and stored in the database. Additionally, if you have authored content within MSCMS, you can use the same publishing API (along with a separate, custom ASP script) to export content from the database into external XML files.

The need that drives the import and export of XML depends on the user's specific business requirements. For example, if you have a great deal of existing content in XML format, you can use the import solution mentioned above to get that content into your MSCMS database. Conversely, if you administer a Web site that must provide certain types of content in XML format, you can use the publishing API along with ASP scripting to save content from your workflow into XML format for redistribution. This section introduces the import process and then briefly describes the export process from MSCMS to XML format.

Importing from XML

The publishing API that is provided with MSCMS is flexible in terms of content access and revision. By using this API, you can add content to and retrieve objects from the MSCMS database. This API allows you to traverse the entire structure of the repository, saving new content in the locations you define.

This API is highly flexible in terms of the information it can access. For example, it can be used to retrieve the entire structure of a document or obtain specified elements from a number of documents. The following steps usually are used to import XML data into MSCMS. More information on the structure of an XML file appears in Chapter 5, "XML Integration with .NET Servers." The steps are outlined in Figure 11.4:

1. Analyze the structure of the XML file, define the placeholder content, and determine which content will become meta data within the MSCMS pages.
2. Define the elements and meta data (for example, custom properties) of the MSCMS template that will be used to create the pages.
3. Create the ASP script that will drive the import process.

Data Flow from XML to MSCMS
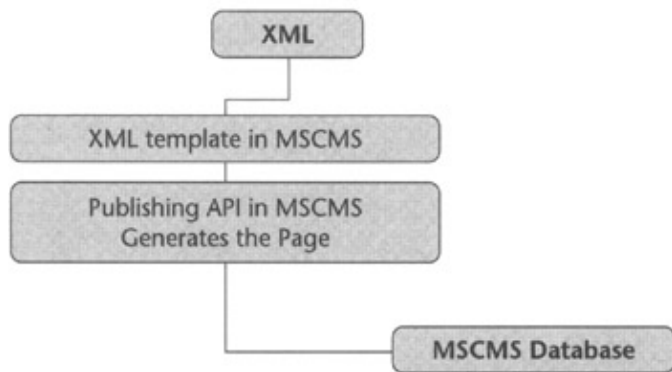
Figure 11.4: Data flow from XML format to the MSCMS database in a manner similar to this depiction.

Before the import process can begin, the content and associated meta data that are to be imported must be structured to assist the automated importation process. The XML file format provides this needed structure by design. Although it does not provide a direct interface, the MSCMS publishing API makes it possible to script the import process. The publishing API allows server–side ASP scripts to interact with the content stored in the MSCMS database. The actual importation of content occurs when a custom ASP script parses the XML elements from multiple source files and then imports the content into MSCMS by using a predefined MSCMS template.

Exporting to XML

An important application of MSCMS involves its ability to manage the dynamic content of a corporate Web site. Through its publishing API, MSCMS is able to export data into many standard file formats, such as HTML, XML, and flat text. XML has become one of the most frequently used formatting languages for distributing content over the Internet; the publishing API allows you to script an information export process that takes MSCMS data and stores those data in XML files. When the content is converted into XML, each XML element is filled in with the content of the corresponding placeholder in the MSCMS data store, and the placeholder attribute is removed.

Workflow

Workflow is a key feature of any content management system. Workflow enables users to publish their own content while ensuring that tasks are completed on time and overall content standards are maintained. Your workflow should be designed to help with the automation of the deployment of content. In performing this automation, take into account the following goals:

- Seamless integration with existing user authentication schemes
- The ability to adapt to existing business processes
- Configurability to enable different levels of security
- Ease of use
- Extensibility

The following sections pertaining to workflow are intended to provide an overview of the considerations for the workflow process. The first section introduces the importance of workflow. This section also addresses the primary security options and considerations related to workflow. The next section describes the tools available for workflow. The last section provides an overview of the configuration for standard workflow.

The Importance of Workflow

One of the advantages of using a content management system is distributed authoring and publishing. It is possible to transfer the control of Web content into the hands of the content experts or content owners. The distributed authoring model effectively leverages the skill sets within a company: People who care about the content can manage it themselves, while the technical staff is not overburdened by having to make content changes to the site. Each group is responsible for its own area of expertise.

However, the distributed authoring and publication model works only when it is combined with a flexible workflow system such as the workflow used in Content Management Server 2001. Using a workflow system ensures that the content on a Web site is:

- Properly approved before it is published
- Published in the right areas of the site
- Published to the right user groups on the site
- Timely, accurate, and relevant

Distributed Web publishing is a new concept for many companies; however, the use of an easily configurable workflow system ensures that a company can take advantage of the benefits of automated publishing without sacrificing control over the validity of its content. To provide a successful implementation of a content management system and facilitate true self–service authoring, workflow features must provide seamless integration with existing user authentication schemes, adapt to a company's business processes, provide flexible levels of security, and above all be easy to use.

User Roles

Content Management Server 2001 offers a flexible workflow process that is integrated with user roles. All content can be subject to review and approval through the workflow process. The following built–in roles are divided into two categories: the roles associated with the standard workflow process and the roles associated with site building.

The standard workflow process roles are as follows:

- **Authors.** Authors create new pages.
- **Editors.** Editors review and modify pages. Editors can approve or decline pages.
- **Moderators.** Moderators provide final approval before a page goes live on the site. Moderators verify publication options such as publication dates, expiry dates, and page names. Moderators also can approve or decline pages.

In addition to these publishing process workflow roles, the following site–building roles are used to control the site–building workflow and access to areas of the site and features:

- **Administrators.** Administrators are superusers who configure the other roles and rights on the site and build the site structure.
- **Template designers.** Template designers are users who create, modify, and approve navigation and page templates. Templates cannot be used until they are approved.
- **Resource managers.** Resource managers are users who add and delete resources, such as videos, graphics, and documents. Authors and editors use resources when they create pages.

Container–Based Rights and Multiple–User Groups

Each standard user role can have many different groups within it, each with different rights. For example, an intranet may have one author group for each department in a company, each with different authoring rights. For instance, only people in the accounting group can publish in the financial services area. The site administrator can create an unlimited number of groups within each role.

The same thing is true for all user roles. By combining the groups or individuals who have permission to work in different areas of the site, it is possible to create a highly secure workflow process. Consider the following:

- **Multiple author groups.** In addition to specifying the areas on the site where each group or individual can publish, administrators specify which templates and resources those groups or individuals can use. For example, only members of the finance group may use the Financial Statements template, and only members of corporate communications may use the corporate logo graphic.
- **Group and individual rights.** Either individual users or user groups can be added to the Content Management Server user role groups. For example, Julie may be the member of a marketing user group with authoring rights to the products area of the site. She also may have individual rights as an editor in the investor relations area of the site. The rest of marketing does not have rights to publish in the investor relations areas of the site.

Combining both individual and group rights provides tremendous granularity for setting publishing workflow and permissions.

User Authentication

Users are authenticated by using .NET Server Active Directory services. Authentication can be set up for individual users or groups. Most sites use a combination of group and individual authentication to achieve maximum control of workflow configuration.

When group authentication is used, whenever a new user is added to a group, he or she automatically will have rights in Content Management Server. If a user is removed from an authentication group, his or her Content Management Server rights are removed. This tight integration with existing user authentication schemes makes Content Management Server workflow fit in with existing business practices.

Workflow Tools

Content Management Server offers easily configurable email notification options and extensive built–in reporting. With these options you can be notified of specific events that are defined in your workflow. The reports allow you to view the pages that are in production and the pages that are awaiting approval:

- Email notification is set up on the basis of the events in the workflow that are used to trigger email. This notification is completely configurable to suit your business requirements. For example, email can be sent to a group of approvers after a new page is submitted for approval, or it can be sent to an author to notify that author that an editor or moderator has declined a page.
- Reporting is used within Content Management Server to provide reports that authors, editors, and moderators can use to check the status of the pages on which they are working. Authors can use the Pages in Progress report to determine page status, including which

pages are awaiting approval and which have been refused approval. Also, the status of every page appears in the Tool palette as a page moves through the workflow process. Editors and moderators can check the Pages to Approve report to view all the pages waiting for approval. They then can approve all the pages at once or check each page individually.

Configuring a Standard Workflow

For many implementations, the MSCMS standard workflow provides ample flexibility. When it is planned properly, the level of granularity of approval is quite high. In introducing a Web content management system, it is sometimes a good idea to keep the workflow fairly simple because a highly complex approval process can create a bottleneck and impede content publication.

When you are planning workflow, you have to consider who will need to publish content on the site and in what areas. What type of approval process does the organization require? In addition to workflow, you want to consider the skills of the content providers. If content providers are new to Web publishing, you may want to restrict the types of content they can publish, where on the site they can publish, and the types of templates they can use. The skills and responsibilities of your content experts should help define the workflow process.

Integration with .NET Enterprise Servers

Microsoft Content Management Server can enhance the solution provided by some of the other .NET Enterprise Servers. By itself MSCMS does little, but when it is used in conjunction with other Web development and publishing products, it is a valuable tool.

You can take the workflow and authoring features of MSCMS and apply them to the Web sites you use for Internet and intranet solutions. Through the creation of templates you can change the look and feel of any Web site quickly. The templates are actually a series of include files that are used to create a general look and feel to a site. Many organizations want all their Web sites (external and internal) to look and feel the same. The templates created in MSCMS can be applied to any of your sites to create a standard presence.

This final section of this chapter focuses on the interaction of MSCMS with some of the other .NET Enterprise Servers. It begins by describing the role of MSCMS in the public Internet arena. This part focuses on the interaction with IIS, Application Center 2000, and Commerce Server. The section then moves to the role of MSCMS in an intranet, showing the interaction with SharePoint Portal Server 2001.

Internet Solutions

Content Management Server can enhance a company's Internet presence. It also can be helpful when it is integrated with some of the other .NET Enterprise Servers that have been described in other chapters in this book. The three products described in this section are IIS, Application Center 2000, and Commerce Server. These three products can provide a fast, secure, and available Internet solution. The following list describes each product from the perspective of what it brings to an Internet solution:

- **IIS.** This is the Web site hosting server. The content is started in an IIS Web site. IIS is the basic level of Web site configuration. At this level you publish pages and handle the incoming requests from the Internet. Application Center 2000 and Commerce Server enhance IIS Web site solutions.

- **Application Center 2000.** Load balancing and synchronization are the key features of Application Center 2000. You can deploy the content across multiple servers to help the site scale to a large number of requests, be available at all times, and provide a consistent connection to your data.
- **Commerce Server.** Web site security and authentication are enhanced when Commerce Server is deployed. You can extend the default authentication options with IIS to include AuthFilter and AuthManager. More information on these authentication options appears in <u>Chapter 10</u>, "Commerce Server."

MSCMS can extend this solution to help control your Web site setting as your presence grows. Among the major weaknesses of sites during the growth stage involve consistency and adherence to standards. Many developers start working on the same projects, and if standards and requirements are not set up front, things can get out of control quickly. Content Management Server solves these problems. By using the template features of MSCMS you can ensure a consistent look and feel to all your sites.

Additionally, the workflow features of MSCMS can be used to create an approval process for the pages that are to be published. Through this process you can help apply standards to your site and verify that the content meets or exceeds those standards.

Intranet Solutions

SharePoint Portal Server 2001 and Microsoft Content Management Server 2001 can be used together to provide a complete solution for content management and publishing for Internet, intranet, and extranet Web solutions. While it is possible to use them for external sites, their real strength is in the intranet arena.

SharePoint Portal Server provides a central point of access to business–critical information and applications through a customized Web–based portal interface combined with powerful enterprise search technology. SharePoint Portal Server also provides a document collaboration and management solution that is especially intuitive for users of Microsoft Office, the Microsoft Windows operating system, and Microsoft Internet Explorer. Content Management Server is an enterprise Web content management product that enables a company to design, publish, and manage highly dynamic and customized Web sites quickly and efficiently. Content Management Server also provides a distributed authoring and publishing environment that utilizes built–in workflow to accelerate the creation of a Web site. Both products offer the scalability and reliability needed for the most demanding Web sites, portals, and Web applications.

Content Management Server and SharePoint Portal Server can be used together to provide unique benefits to medium–size and large businesses. This section describes how SharePoint Portal Server and Content Management Server can be used to accomplish the following goals:

- Increase the efficiency of content publication to intranet portals and Web sites
- Reduce content duplication and increase the consistency of published information
- Increase the productivity of knowledge workers who use portals and publish Web content

This section introduces SharePoint Portal Server and then addresses the ability to integrate the two products.

SharePoint Portal Server 2001

SharePoint Portal Server 2001 is a flexible portal solution that integrates search and document management with the tools you use every day. SharePoint Portal Server combines enterprisewide search, document collaboration, and content aggregation to enable a company to deploy intranet portal sites for teams, business units, and the entire organization. SharePoint Portal Server works with Internet Explorer, Microsoft Office applications, and Web browsers to help a developer create, manage, and share content. Intranet portal sites built with SharePoint Portal Server are the simplest way to access business information. The features of SharePoint Portal Server include the following items:

- **Publishing to an intranet portal site.** An intranet portal makes it possible for users inside or outside an organization to search for and gain access to critical business information regardless of format.
- **Searching across multiple locations.** Users can search a variety of sources, including SharePoint Portal Server workspaces, Web sites, file systems, Microsoft Exchange Server folders, and Lotus Notes databases.
- **Accessing a document on the basis of user roles.** Role–based security helps you control access to documents and other types of content.
- **Tracking versions of multiple documents.** Check–in and check–out features ensure that only one person works on a document at a time, and versioning makes it possible to maintain archival copies of content for reference and recovery.
- **Reviewing and approving.** Document routing enables you to control when a document is ready to be published to an intranet portal site.

More information on SharePoint Portal Server is given in Chapter 13, "SharePoint Portal Server."

Using SharePoint Portal Server 2001 and Content Management Server 2001 Together

This section describes three scenarios for using SharePoint Portal Server and Content Management Server together. These scenarios allow you to perform the following tasks:

- Link the publishing processes for Web sites and intranet portals to shorten publication cycles
- Share content between Web sites and intranet portals to decrease duplication and increase consistency

These scenarios use the products to their best advantage: Content Management Server for managing the design, publication, and management of Web sites and SharePoint Portal Server for providing portal services, enterprisewide search, and a document–collaboration environment that includes document management.

Linking the Publishing Processes for Web Sites and Portals

Distributed authoring and publishing is one of the advantages of using Content Management Server and SharePoint Portal Server. In most companies a small number of knowledge workers create most of the documents that are published to Web sites and portals. The distributed authoring model keeps the control of Web content in the hands of content experts and owners, enabling the people who create content to manage it themselves and freeing technical staff members from the burden of making changes to content with which they may be unfamiliar.

Good review and approval processes are essential for achieving a distributed authoring and publishing environment. Using the review and approval features of SharePoint Portal Server and Content Management Server ensures that a company can take advantage of automated publishing to Web sites and portals without sacrificing control over the content being published.

Both SharePoint Portal Server and Content Management Server include review and approval tools. SharePoint Portal Server has a document review and approval process for publishing content to intranet portals. Content Management Server includes a formal workflow process that ensures that the content on a Web site is approved before it is published, is published in the correct areas of the Web site, is available only to those with the correct access rights, and is published in a timely manner.

In this scenario, using SharePoint Portal Server and Content Management Server together enables you to set up a connected two–tier workflow process for creating, approving, and publishing content, thus shortening the time needed for the content creation life cycle. The first tier is the workgroup collaboration and approval phase, in which documents are created, stored in SharePoint Portal Server, and then published to intranet portal sites.

The second tier is the Internet publishing phase, which begins when a document stored in SharePoint Portal Server is promoted to Content Management Server. Once it is promoted, the document is passed through the Content Management Server approval process and published as part of a Web site. If that document subsequently is changed in SharePoint Portal Server, it can be updated automatically in the Content Management Server repository and on any published pages on which it is included. You should consider the following items when linking pages between the two products:

- When a document is first promoted from SharePoint Portal Server, a content author using Content Management Server chooses which page of the Web site it will be appear on. When the same document subsequently is modified in SharePoint Portal Server and again promoted to Content Management Server, it is updated automatically because the document location already has been determined. Content Management Server searches its repository for the document's unique identifier and replaces it with the newer version of document promoted from SharePoint Portal Server.
- The first time a document is promoted from SharePoint Portal Server to Content Management Server, it is subject to the complete Content Management Server workflow process. For subsequent times when a document is promoted, Content Management Server can be configured either to subject the document to the complete workflow process again or to update the document in the Content Management Server repository automatically without further approval. In this way, the workflow linkage between the two products can be adapted to the practices of the company.
- When the publication date of a document that already has been promoted to Content Management Server from SharePoint Portal Server changes, a script is triggered that causes Content Management Server to upload the new file. This script, which runs on SharePoint Portal Server, promotes the document to Content Management Server.

Sharing Content between Web Sites and Intranet Portals

Often the same content will be published to both Web sites and intranet portals. The difference between publishing content to an Internet site and publishing it to an intranet portal should be transparent, especially to a content author or business user. Using the same content on both Internet sites and intranet portals is easy when SharePoint Portal Server and Content Management Server are used together. Using these products together reduces duplication of information and

allows you to maintain consistency between the content published on your intranet portals by SharePoint Portal Server and the content published on your Web sites by Content Management Server.

In this scenario, a document is first stored in SharePoint Portal Server. The document probably will be published to one or more intranet portals created by using SharePoint Portal Server. The document can be retrieved from SharePoint Portal Server by Content Management Server and included as content that Content Management Server publishes to Web sites. Content Management Server also can publish content retrieved from a search performed by the SharePoint Portal Server search engine. Another way for Content Management Server to access documents in SharePoint Portal Server is by using a Content Management Server template that includes a link to the document or directory. A template can access a specific directory on the basis of the characteristics of the directory, such as its location.

A document retrieved from SharePoint Portal Server and published by Content Management Server is displayed as an attachment file that can be downloaded from the Web site. In addition, properties of the document such as author, title, and summary can be published by Content Management Server. The downloadable document and the information in the document's property can be published on the same page. Note that documents stored in the SharePoint Portal Server database are available to be published on Content Management Server sites only when their status is set to "published." This feature allows a content creator to determine which draft of a document stored in SharePoint Portal Server is to be published by Content Management Server.

Using Content Management Server and SharePoint Portal Server together also keeps the information published on a company's Web sites and intranet portals secure. This security is essential when content contributors want to publish only a subset of the content they have created.

Content Management Server authenticates all visitors to a Web site by using security features of the Windows operating system. However, access to individual areas of a Web site and to the content on individual pages is controlled by Content Management Server. Page–level security is an important feature of Content Management Server. For example, Content Management Server can allow all the employees in a company to see the overview portion of a financial statement on a Web page but ensure that only members of the accounting group can see a link to the entire statement.

Review Questions

1. What is the role of Content Management Server?
2. What is the purpose of Site Builder?
3. What are the main design goals of Content Management Server?
4. What can the flexible workflow features be used for?
5. What is the role of MSCMS role in Internet development?
6. What is the relationship between Content Management Server and SharePoint Portal Server?
7. What is the advantage of sharing content between Web sites and intranet portals?

Chapter 12: BizTalk Server

Overview

Integration is a challenge that many businesses face. The two primary issues in this area are the need for cross–platform integration and for business–to–business scalability. Early solutions to these problems were resource–consuming and expensive and did not necessarily perform well. Specifically, they did not provide a platform for electronic data exchange that would make a business accessible from anywhere at any time, reduce process time, help consumers make better decisions faster, manage and share knowledge effectively, and manage dynamic business relationships efficiently. Many organizations struggle to achieve the ability to integrate business applications and business processes. BizTalk Server is an integration product. At its core it is a tool to integrate applications, databases, and processes.

BizTalk Framework 2.0 is a business framework that addresses integration problems efficiently because it is based on XML and therefore is completely platform–independent. The central idea for this framework is XML–based message transmission. BizTalk Server 2000 represents Microsoft's implementation of the BizTalk Framework. A developer can take advantage of his or her XML knowledge and use it to help integrate his or her applications.

There are many reasons why the BizTalk Framework can be used successfully. One important factor is that message passing using XML and XML–based schemas meets many requirements for interactions between systems or organizations. As organizations try to integrate dissimilar systems and automate business processes, XML integration with the BizTalk Framework can be advantageous. Some of the advantages of XML and BizTalk integration include the following:

- A flexible, universal language to describe and package information, allowing both structured and unstructured forms of the information
- A flexible mechanism for defining transformation rules for converting information from one format to another as needed
- Platform–independent communication protocols that handle interactions between applications or organizations
- Platform–independent security mechanisms to protect privacy and the integrity of the data exchange

This chapter introduces BizTalk Server, beginning with its architecture. That architecture contains many components that supply enhanced support for .NET services. That section includes a description of some of the components that make up BizTalk Server and also describes its integration with XML in more detail.

The next section deals with the features of BizTalk Server for .NET development. Several of the BizTalk Server features, such as transfer maps and messaging services, are used only to integrate two dissimilar applications. BizTalk Server features are supplied to enhance enterprise development with Microsoft products.

The chapter concludes with a section on .NET business solutions with BizTalk Server. This section describes the role of BizTalk Server in application integration solutions.

BizTalk Features for .NET Services

BizTalk Server can handle complex business communication and integration both within an organization and across organizations. The components and features of BizTalk Server are centered on the concept of integration. The BizTalk Server environment and architecture consist of multiple components that can be categorized under the three layers shown in Figure 12.1.
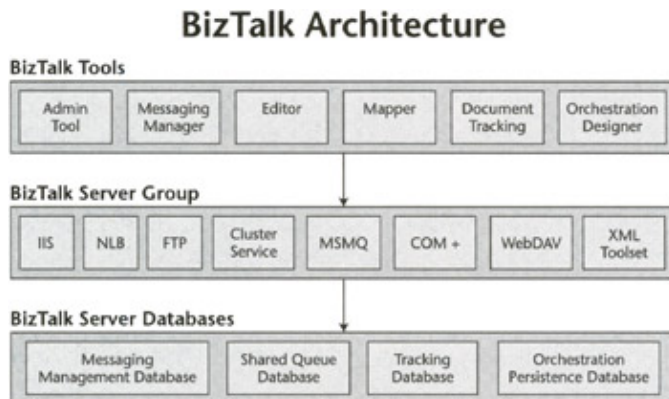

Figure 12.1: The BizTalk architecture includes multiple components that can be grouped into three layers: BizTalk tools, BizTalk server group, and BizTalk server databases.

The following sections describe the three layers of the BizTalk Server architecture. Each layer is presented in a manner that reflects its relationship with .NET services. Each layer has the core components, which are described in more detail. The section then discusses the integration of XML and BizTalk Server and concludes by describing the relationship between SOAP and BizTalk Server.

BizTalk Tools

BizTalk tools are the administrative tools that are used to manage BizTalk Server. These tools can create a business solution. Each of the tools integrates with a BizTalk server group and an SQL Server database, as described here:

- **BizTalk Administration.** This graphical user interface (GUI)–based tool manages queues, transports, and services. This is the core configuration tool for server–level configuration.
- **BizTalk Messaging Manager.** This component allows you to create trading relationships. With this tool you can define trading partner objects and their characteristics and define the relationships between those objects and the objects you use so that you can exchange documents with partners over the Internet. For example, for some trading partners you may need to use Secure Sockets Layer (SSL) certificates, while for others that may not be necessary. One advantage to using the Messaging Manager is that a user does not need programming experience to use it.
- **BizTalk Editor.** This is a GUI–based editor for defining XML business document schemas. By using this tool, you can create document specifications by defining records and fields by hand. You can import well–formed XML document instances. Internally, all the documents are represented by using the XML schema, but this is transparent to the user.
- **BizTalk Mapper.** BizTalk Mapper allows you to take an XML document such as a purchase order from one application and map it to a corresponding document in another application. The files that the mapper generates are in the Extensible Stylesheet Language Transformations (XSLT) format, which is a World Wide Web Consortium (W3C) standard. In addition, the mapper can map to non–XML formats. For example, you can map documents between electronic data interchange (EDI) and XML formats or between the XML format and a flat file. The mapper can even map between non–XML formats. For example, a user can

create a mapping between the EDI format and a flat file.

- **BizTalk Document Tracking.** This GUI−based tool tracks message and schedule activity. It is handy when you want to analyze and troubleshoot business processes.
- **BizTalk Orchestration Designer.** This GUI−based tool is convenient for designing and building dynamic distributed processes. This is an easy−to−use BizTalk application design environment that is useful to business analysts, information technology professionals, and developers. A developer can use this tool to design a business solution.

BizTalk Server Group

The middle component layer in the BizTalk architecture is the BizTalk server group. The components in this layer are not necessarily exclusive to BizTalk Server. Most of the items in this layer are commonly referred to as .NET services. BizTalk Server is highly dependent on .NET services, which provide BizTalk Server with the ability to perform its necessary functionality. These middle layer components include the following:

- **Internet Information Server (IIS) 5.0.** IIS provides components necessary for scalable HTTP and ASP.NET processing.
- **Network load balancing (NLB).** This optional component works with IIS to provide even more scalability. The load on servers can be balanced by this .NET Server or the Application Center service. Load balancing is covered in more detail in Chapter 9, "Application Center 2000."
- **Cluster service.** This optional service provides fault tolerance and greater dependability for shared SQL Server databases.
- **Message Queuing (MSMQ) and COM+.** This component provides Component Object Model (COM)−based integration and MSMQ receive functions and transports.
- **Web Distributed Authoring and Versioning (WebDAV).** This component provides distributed access to maps and documents schemas.
- **Simple Message Transport Protocol (SMTP) functionality.** This component transmits Internet mail.
- **File Transfer Protocol (FTP) functionality.** FTP is used to transmit files.
- **XML toolset.** Although this is not a .NET Server component, the XML toolset conceptually belongs on this level. It provides support in the form of a validating parser and contains other important processing tools. This toolset provides the medium for application communication and validation.

BizTalk Server Databases

The last component layer consists of the BizTalk server databases. These databases are stored in SQL Server. BizTalk Server is highly dependent on SQL Server. Access to an SQL Server is needed to store the following list of databases:

- **BizTalk Messaging Management database.** Manages groups and servers. The servers share the same configuration information that describes the document processing logic.
- **BizTalk Shared Queue database.** Manages documents and interchanges for all the servers in a group. This is where the four SQL Server queues described earlier reside.
- **BizTalk Tracking database.** Stores information about document and interchange activities and for checking the status of documents.
- **Orchestration Persistence database.** Stores the information about business orchestration.

XML Integration with BizTalk Server

By far the most successful integration solution available today is XML–based message passing between different platforms. XML is a platform–independent standard that describes data to be transmitted between computers regardless of the application or operating system. In other words, XML offers a common document representation that everybody can understand. There is no need to use interpreters or learn another language (except for some very basic phrases) because XML offers a common metalanguage that you can use to communicate.

The BizTalk framework supplies the generally agreed–upon set of basic assumptions mentioned previously (functionality and interface guidelines) and also provides the flexibility to extend them as needed. In terms of the business objectives discussed in the previous section, BizTalk supports those objectives while attempting to minimize the risk factors by allowing businesses to use the infrastructure they already have. There is no need for costly and error–prone modifications.

To use a human analogy, if you have always spoken French, you do not have to learn German but can continue doing business in French as you always have. And your German trading partners can happily do business with you—in German. Thus, if your current applications work, you know that the solution will work as well. The operational requirement is also satisfied because you can extend your French vocabulary as much as you like, and as long as your partners understand your tags, they can understand your documents. Also, the cost of implementing this solution is low compared with that of earlier solutions. It is French throughout for you. Similarly, if your partners use German, they do not need to learn French to talk to you. In fact, if you use XML, you do not even have to modify your existing code. All you have to do is rely on the BizTalk Framework to handle the documents for you.

SOAP and BizTalk Server

Although SOAP is not a part of BizTalk Server, the two are often categorized together in speaking about the use and integration of XML. The SOAP protocol defines the format of standard XML messages that communicate among systems. Because the message format is standardized and is based on the XML standard, SOAP can communicate among multiple computer architectures, languages, and operating systems. SOAP enables a new class of applications, called Web services, that expose services in a standard way so that application developers can create new applications by putting together services from many different sources on the Web.

SOAP is designed to be simple and extensible. It defines a message–processing model but does not itself define any application semantics, such as programming model and implementation–specific semantics. Although SOAP may be used as a foundation for building complex systems, most features from traditional message systems and distributed object systems are not part of the core SOAP specification. A SOAP message is based on XML and contains the following parts:

- The envelope is the top–level container representing the message.
- The header is a generic container for adding features to a SOAP message in a decentralized manner. SOAP defines attributes to indicate who should deal with a feature and whether understanding it is optional or mandatory.
- The body is a container for mandatory information intended for the ultimate message receiver. SOAP defines one element for the body to report errors.

SOAP is not exclusive to BizTalk Server. It is included with extensions for Visual Studio, Windows XP, and .NET Server and is beneficial in extending the functionality of XML Web services. BizTalk

Server is built on the foundation standards that are provided by XML and SOAP and can take advantage of those protocol standards. SOAP is defined by the SOAP standard, which is available at www.w3.org/TR/SOAP.

Features of BizTalk Server for .NET Development

BizTalk Server has several core features that define its overall functionality. This section of the chapter describes several of these core features. This section outlines the features of BizTalk that are helpful in .NET development, presenting those features as they relate to .NET application development. The details of implementing each of these features can be found in the BizTalk Server product documentation and at http://www.microsoft.com/.

Document Specifications

Document specifications are a feature that enhances business–to–business (B2B) application solutions. An organization can create a business solution for document sharing between organizations. Having formal document specifications provides important benefits. First, if two or more trading organizations agree on a particular specification, they can exchange documents without worrying about whether the trading partner can process those documents. This is the case because all documents that conform to one specification can be processed in exactly the same way.

The other benefit has to do with document creation. Once you know the specification, you can automatically verify whether a document conforms to your specification. This makes it easy to transfer documents between organizations without the possibility of the other entity not complying with your specifications. This section provides an introduction to document structure and the specifications that are immediately available in BizTalk Server. It then describes the hierarchical approach and data relationship management of document specification. Finally, it discusses the validation of documents through BizTalk Server.

Overview of Document Specification

Document specification is a requirement of BizTalk Server. The specification is used to define the document structure. Several existing industry standards describe various common business documents.

A user can create document specifications from BizTalk Editor, the primary management tool for the administration of BizTalk Server. BizTalk Editor makes available the industry–standard document specifications. Any of these standards can be used as is or modified to suit individual needs. This section describes the three standards: two standards supported by EDI (X12 and EDIFACT) and one supported by XML. XML has become the most successful solution. It also is used as a .NET service, and so many of the concepts in XML are used throughout the .NET Enterprise Servers.

In addition to these built–in standards, you can create your own document specifications that are based on one of the standards. You also can create a document specification that is independent of these standards. The following list provides an overview of these standards:

- X12 standards generally are accepted in the United States by both government and business. They are less well known elsewhere, particularly in Europe, where UN/EDIFACT dominates. The X12 standard consists of items called transaction sets. Each transaction set represents a business document. Currently, there are close to 300 available transaction sets supporting various business areas, such as communications and controls, product data,

finance, government, materials management, transportation, purchasing, industry–standard transition, distribution and warehousing, and insurance. The beginning of this section refers to X12 standards, not to a single standard. This is the case because over time there have been many releases of the standard, as needed. A release might add new transaction sets or modify old ones. In fact, in some years there have been as many as three releases of the standard. To complicate matters further, the releases are not upward– or downward–compatible. Some of the standards can be found in the \Program Files\Microsoft BizTalk Server\XML Tools\Templates\X12 folder, which contains four standards and shows how the schemas evolved over time. The schemas are not guaranteed to be compatible between versions.

- The United Nations rules for Electronic Data Interchange for Administration, Commerce, and Transport (UN/EDIFACT) constitute one of the oldest standards still in use. The standard emerged in the United Kingdom in 1974, when the British delegation to the United Nations described a proposal "for providing standard ways of representing information in messages passing between partners in international trade." The goal of the proposal was to establish rules for structuring data in such a way that the standard was independent of system and media constraints and also was accessible to users. The standard has been called "a nightmare of the paperless office" because it is cumbersome to use. The formal description of the standard consists of 2,700 pages because its creators attempted to design a standard that would apply to all the forms of documents a business user would ever need. EDIFACT recognizes two actions that a business document will go through: processing and transmission. The standard attempts to address both of these transactions while also providing mechanisms for document verification. Currently, there are six versions of the EDIFACT standard that are not guaranteed to be compatible with each other.
- XML Schemas can be used within the BizTalk Framework. These schemas are far less cumbersome and allow you to customize your specifications. BizTalk.org is the organization whose goal is to popularize the framework by helping businesses share the document specifications they use for B2B document exchange. It is a far less formal organization than either ANSI (the organization that has defined X12) or UN/EDIFACT. In fact, there is no single standard to which businesses conform. Instead, each business can register its schemas with BizTalk.org or locate and use schemas that other businesses already use. It also is possible to register a schema in a secure area so that only one's trading partners can access it.

Hierarchical Data Structure

To create a document specification, one must think about how the data in the document are organized. The document will contain multiple data items. Those data items most likely are related in some way; otherwise they probably would not be in one document. However, some relationships are closer than others. For example, in a purchase order sent by a client to a vendor, one would expect to see information such as the name of the client, the address, the product name, and the product quantity. While all those pieces of information pertain to the purchase order, they differ in their relationships to one another. The street address of the client is closely related to the city in which the client is located, but the city is loosely related to the quantity of the product being purchased. However, the product quantity is closely related to the product name because both help determine what the purchase item consists of.

Usually there are many such relationships in a document. A document specification describes those relationships formally by organizing the items into a hierarchy in which the closely related items are a short distance away from each other, and the loosely related items are farther away from each other. A document therefore can be viewed as a hierarchy of nodes with a root node at the top. The nodes below the root can be records or fields. The lowest level of the structure consists of fields,

which relate directly to data contained in the document itself. This is exactly how BizTalk Editor represents document specifications.

Figure 12.2 shows the simplified structure of a purchase order for the client–vendor relationship mentioned earlier. At the root is the Purchase Order node. The purchase order consists of three items, which are child nodes of the root node: Client Name, Client Address, and Item Ordered. Client Name is a field. Client Address is a record that contains three fields: Street, City, and Zip. The Item Ordered node is a record that contains two fields: Item Name and Quantity. In a real–life scenario the purchase order would consist of many more fields.

Figure 12.2: A client–vendor purchase order model that displays the hierarchical nature of a document specification.

Document Validation

In BizTalk Editor it is possible to validate both document specifications and documents themselves. The validation of document specifications involves checking the form of the specification. A specification must be a well–formed XML document and must conform to any specified constraints.

Validating a document involves checking the adherence of the document to the specification. This includes checking the document structure to make sure that the defined records and fields are in proper relation to other records and fields. The validation ensures that the required records and fields are indeed present in the document. In addition, the data types of fields are checked, and adherence to the specified number of looping records is enforced.

In most cases BizTalk Editor is used to create document specifications. Such specifications are guaranteed to be well–formed XML documents. In addition, BizTalk Editor has other built–in safeguards that ensure that a document created within the BizTalk Editor environment will validate without any problems.

However, not all document specifications will be problem–free. For example, you may agree that your trading partner will supply you with the document specifications based on your common needs. In such cases you have no guarantees that the specification is valid. Also, it is possible to create document specifications outside the BizTalk Editor environment. After all, a document specification is simply a well–formed XML document. Therefore, you can use any text editor to create document specifications. If you do this, the safeguards built into BizTalk Editor will be unavailable, and the specification may have problems.

You need to make sure that the specification itself is valid before you can validate any documents against it because validation of documents against an invalid specification has no meaning. Follow these steps to validate a document specification:

1. Open the BizTalk Editor tool from the BizTalk Server 2000 program group.
2. Open the specification you created from the BizTalk Editor tool.
3. From the Tools menu, choose Validate Specification, as shown in Figure 12.3.
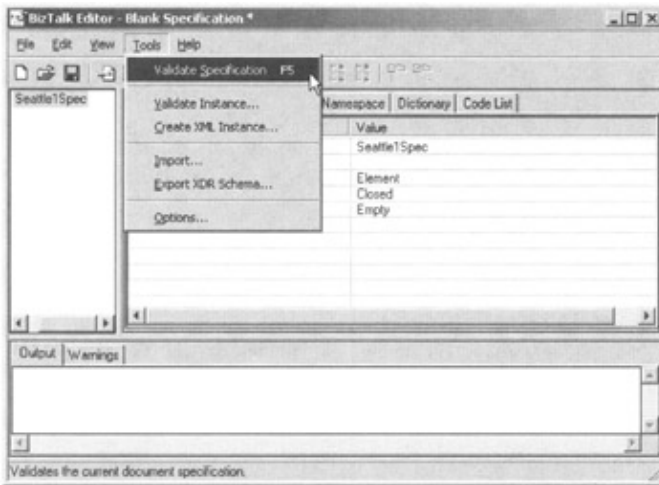
Figure 12.3: Document specifications should be validated to resolve any errors related to the document.

    4. The Warnings tab displays any errors and warnings found.

After validating the specification, you will need to resolve any errors or warnings to ensure successful use of the document.

Maps

As you exchange data from business to business, the document has to be in a fixed format. Many businesses transfer data from database to database by using Object Linking and Embedding Database (OLE DB) or Open Database Connectivity (ODBC). This is a fast method of data transfer, but it has some disadvantages:

- You are limited to the functionality of the driver. You do not customize the way in which you communicate between the systems.
- You need to update the driver as the application that is integrating the systems upgrades. Often, the drivers get out of date, and you start receiving ODBC data access errors.
- The transfer does not go over a standard Internet port. XML typically transfers data over port 80, a common port for Internet traffic. It is much more difficult to transfer data over the Internet using ODBC and OLE DB.

XML and BizTalk Server can help overcome these disadvantages. In many cases the format of one organization does not directly match the format of the other organization, and there is a need for maps. A map is a custom mapping (transformation) of data. You are defining the method of the data that will be transferred. In addition to the transfer you are defining modifications that should be made along the way. For instance, one organization may refer to the customer's last name as LName, and the other organization may refer to it as LastName. These two data elements have to be mapped to each other. Creating maps provides the following advantages:

- It helps business integration between organization units or trading partners. By applying maps to documents, trading partners can exchange business documents without either party having to change its processes. This is advantageous in several typical business scenarios. For example, if a new trading partner enters the exchange, it probably will have automated some business processes already that require specific forms of documents. However, the form that the other trading partners use probably will be different. Without maps, the new trading partner will have to alter the process to accommodate the new document formats. If maps are used, no alterations are necessary, and this allows the new partner to transform

the documents already in use into a form that the trading partners can use. Another common situation in which maps are helpful occurs when two businesses merge. The two businesses almost certainly use incompatible document formats. By using BizTalk Mapper, they can process each other's documents without much programming effort.

- It allows you to use one document containing data that are interesting to several different users and to create new documents that are based on this document (and therefore on the same information) but that have different forms to meet various business needs. This is useful, for example, when a user accesses a database or receives raw data from another unit or organization and then has to transform the resulting raw data into several formats for different users.

- It allows you to automatically edit information you are presenting in the destination document. The destination document does not have to contain all the data that the source document does. If for some uses you need only a subset of the data, you can create a map that picks out the necessary information from the source document and discards the rest. For example, you may be interested in only a specific item in a purchase order that contains multiple items. Similarly, you can create summaries and aggregates of the data, filter the data according to predefined criteria, find the maximum or minimum data item, and so on, depending on the specific user's needs.

The BizTalk Mapper tool is used to create and manage your mappings. Refer to the product documentation for details on implementing transformation mappings.

BizTalk Messaging Services

BizTalk Messaging Services is the Microsoft BizTalk Server 2000 component that manages the exchange of documents between trading partners. Once the document specifications have been determined, Messaging Services coordinates the communication. This is essential to making sure the data get to their destinations.

Messaging Services consists of objects that are used to describe trading partners, participating applications, and channels used to exchange data. The basis of BizTalk Messaging Services is the channel, an object that directs the document exchange. Other objects play supporting roles, providing additional information and performing other supporting functions. This section describes the Configuration Object Model and then describes the general order in which the objects that support the messaging services should be created. The details of each step are provided in the product documentation. Several examples of Messaging Services are available at Microsoft's Web site at http://www.microsoft.com/.

Messaging Object Model

The underlying structure that makes document processing possible is the BizTalk Messaging Configuration Object Model. You can use BizTalk Messaging Manager to configure the objects in the BizTalk Messaging Configuration Model or configure them programmatically. The following objects are available in the BizTalk Messaging Configuration Model:

- **BizTalkConfig.** Used for creating and configuring other BizTalk Server objects. The objects that can be created by using this object include messaging ports, channels, and document specifications.
- **BizTalkOrganization.** Contains the attributes of the organizations.
- **BizTalkDocument.** Contains the attributes of the document specification.
- **BizTalkEnvelope.** Stores the attributes that identify the format of the envelope for a document. This object is used mainly with interchanges and documents that use the EDI formats, such as X12 and EDIFACT.

- **BizTalkPort.** Stores information about the destination of a document submission.
- **BizTalkPortGroup.** Stores information about individual messaging ports. This object is used for configuring and using groups of messaging ports into distribution lists to move a document to multiple destinations.
- **BizTalkChannel.** Stores information about the configurations of the source entity and binds the source entity to the specific messaging port. It also can specify a map that is used for transforming the inbound document into the form specified for the outbound documents.

Creating Messaging Objects

Messaging Manager is a convenient interface for managing Messaging Services objects. In addition to creating and deleting objects, you can configure objects to meet your specific needs. However, BizTalk Messaging Services imposes certain restrictions on the order in which objects are created. The order for object creation is as follows:

1. Create and configure the home organization. The home organization is the representation of your organization.
2. Create other organizations with which you will interact.
3. Create your documents.
4. If needed, create envelopes. You can perform this step together with the previous step, creating a document and then creating the envelope for that document before you move to the next document.
5. Create messaging ports.
6. Create channels for the messaging ports.
7. Create a distribution list, which is the group that will receive the message.
8. Create channels for the distribution lists.

While this sequence is recommended for creating objects, the order is not completely fixed. Certain steps can be performed in a different order if you keep in mind that you cannot reference objects before they are created. You can exchange the order of steps 6 and 7, for example, because both the channels and the distribution lists depend on messaging ports but not on each other. There is more detail on object dependencies later in this chapter.

BizTalk Orchestration Services

BizTalk Orchestration helps you create self–documenting business designs and makes it possible to implement the designs directly by adding components and defining the interactions. BizTalk Orchestration Designer is a tool for creating and compiling business processes. The document design that is used must fulfill two requirements. The first, core requirement is to express the business process in a graphical, easy–to–understand form. The second requirement is to make the design easy to extend and modify. In particular, the tool should make it easy to add new products to the design and make it just as easy to add new partners as it is to add hardware components to a computer. The goal of orchestration is to add trading partners in a plug–and–play way.

This section defines the workflows within Orchestration Services. The section then describes the environment provided to create and manage the Orchestration Services. It then defines the orchestration process. Finally, this section provides the benefits of using BizTalk Orchestration Services.

Workflows

One of the central ideas in this process is workflow, which is probably intuitive, although it is worth defining in a more precise way. A *workflow* is a methodology that can be used to design procedures

in a business environment, such as within an organization. You can use the designs you have created to improve the procedures and automate them.

The basic elements of a workflow are the steps of the procedure it designs and the dependencies between them. You also can indicate which *entities* (also called *roles;* these are persons or applications) are responsible for which tasks, how long each task is expected to take, and what happens if a deadline is not met. In addition, you can specify the routing of items such as documents. Depending on the specific implementation of the workflow, you can make the workflow engine responsible for routing or have the participants take some or all of that responsibility. With BizTalk Orchestration Designer, you use the latter method, and the participants determine how the documents and control are routed.

After the business process that shows all the messages that are to be exchanged has been defined, it can be compiled into an executable flow. In BizTalk Orchestration Designer such a compiled process is called an XLANG schedule.

The BizTalk Orchestration Designer Environment

BizTalk Orchestration Designer offers a formal language that can be used to describe business processes and a way to specify how documents flow within the business processes. The graphical language used by BizTalk Orchestration Designer makes it easy to illustrate and understand how business processes flow. Using the appropriate shapes, you can express the beginning and the ending of a business process, the sequence of processes, branching, and process concurrency. Once you are satisfied with the design, you can bind the business process elements to specific ports to show the flow of documents between the process you just defined for the organization and the outside world. A port specifies a location from which the data can be inputted into an XLANG schedule or outputted from it.

BizTalk Server uses XLANG, a special XML–based language, to describe business processes. The first step in creating a formal business description is to create an XLANG schedule drawing. A developer creates this drawing by using various tools and connecting them through the use of the graphical language. Once the drawing is complete, you indicate how it is to be executed by binding your processes with implementation ports. Finally, you compile the XLANG drawing to create an XLANG schedule or an XML–based description of the workflow. You then test and debug the schedule to get it ready for production.

Note The compiled XLANG schedule is significantly different from any other compiled entity. One cannot simply run an XLANG schedule on a server because typically there is no single server on which to execute the entire schedule. Instead, it can be thought of as a binding contract between an organization and other organizations, almost like a legal document that describes how the parties interact and what the result of those interactions are.

Benefits of BizTalk Orchestration

Business processes can be quite complex even in relatively small organizations. Processes also change over time as an organization grows and acquires new business partners. Processes must accommodate the needs of not only the home organization but the partners as well. Consider the following benefits of BizTalk Orchestration:

- **Provides an explicit business model.** Every organization follows a business model. Unfortunately, not every organization has a formal or even an informal description of that model. In fact, for many businesses, the business model lives inside people's heads. Many businesses have tried to work around this problem by creating documentation about the

business model. In the ideal world documentation accurately reflects business processes and is updated every time there is a change. Unfortunately, reality is very different. Many people do not understand the value of keeping the documentation up to date, become too busy and simply forget, or leave before they can make the necessary changes. There are countless reasons why documentation does not get updated, but the result is always the same: the loss of important business intelligence. BizTalk Orchestration Designer bypasses this problem by rolling the product and the documentation into one unit. BizTalk Orchestration Designer makes it possible to express the way in which business processes relate to each other within an organization. The advantages of having a formal description of business processes are obvious. An explicit business model can be consulted when changes are needed or can be used to figure out how the processes are done so that one can program against this model. Then, when you make changes to the way your business works, you do not need to perform the extra step of documenting them.

- **Adds business partners easily.** Many earlier methods to describe business processes offered the possibility of achieving business document flow between trading partners. However, this usually was done in pairs. That is, it was possible to create, describe, and program against a model that described interactions between an organization and one specific business partner. Adding another partner was not easy. In fact, it was necessary to create additional pairings for each partner. This approach suffered from the combinatorial explosion problem: For three partners the overall solution requires three pairings; for four, six; for five, ten; and so on. With BizTalk Orchestration Designer it is not necessary to create a special solution for an additional trading partner because BizTalk is designed for multipartner interactions.

- **Unifies the analysis and coding stages.** Traditional attempts at automating business processes consisted of two stages: design and coding. The design stage consisted of analyzing the existing business processes, and the result of the analysis was some sort of diagram, typically a workflow or interaction diagram, that described the processes. During the analysis stage steps might be taken to streamline and improve business processes, but overall the focus was on describing the existing situation, not on improving it. Analysis usually was performed by business analysts who did not necessarily have a programming background. The coding stage consisted of analyzing the diagram produced in the analysis phase and creating applications that followed the model. The programmers responsible for the deliverables of this stage did not necessarily have a business background. BizTalk Orchestration eliminates the gap between the business analysts and the programmers who were responsible for the two stages. Now one team or even one person can perform the tasks that once were done in distinct stages. The tools BizTalk Orchestration Designer supplies are integrated, and so the process no longer has to consist of two stages. The distinction between a business analyst and a business programmer–implementer also has been blurred. The result is an environment that allows the rapid development of business solutions.

- **Updates the business process.** Another big advantage of using BizTalk Orchestration is its flexibility. Business processes, or models, are not constant; they change over time. Some changes are small. For example, you might add a new product to sell that is related to the product your business is already selling, and implementing the changes necessary to process the new sales is not a complex task. However, some changes are substantial or even fundamental. For example, acquiring a company and integrating its processes with the processes already in place require substantial effort. Many changes fall somewhere in between. But whatever the change, it is far easier to change the business model and implementation if the original situation has been orchestrated. Adding or removing processes is easy with BizTalk Orchestration Designer.

- **Supports short–running and long–running processes.** With BizTalk Orchestration Designer you can model all kinds of business processes, from simple to complex, ranging in

duration from seconds to weeks or even months. For example, you can orchestrate the process of ordering and delivering online an electronic version of a book, a process that can be performed in a matter of seconds. After all, if the electronic version of the book a customer wants to buy already exists online, it can be downloaded to the customer as soon as a credit check has been completed. One can just as easily orchestrate a process involving the ordering, producing, and delivering of an airplane, a process that will take months to execute. The credit check for this process probably will take longer, the plane takes weeks to build, and some parts may have to be ordered from the supplier, who may need time to manufacture them. The testing also takes time, and the delivery is not as easy as is the delivery of goods electronically. However, the fact that both of these processes can be orchestrated in the same way constitutes a big advantage of using BizTalk Orchestration Designer.

- **Treats process and schema as commodities.** With orchestration, a new and interesting possibility arises. Because the process is now presented in the form of an object, that object can be published. Similarly, document schemas now can be treated as commodities. In other words, a business entity can publish its business model, and trading partners who want to do business can be required to use that model.
- **Integrates different technologies.** Orchestration makes it possible to integrate dissimilar applications into a business process. This is the case because of flexible binding. An XLANG schedule communicates with the outside world by using implementation ports. The role of ports is to bind XLANG schedules to specific technologies. However, the details of the implementation of these ports are not visible from within the XLANG schedule. It is not the schedule but the port that implements a specific technology. If you later want to use a different technology, you do not have to change the XLANG schedule. All you need is a new binding.

Security Considerations

Business–to–business integration poses the unique challenge of reconciling two conflicting goals: data protection and business integration. On the one hand, it would be extremely unwise not to protect data against an outside intruder. After all, business data are a valuable commodity. This is one reason business databases are tempting targets for attacks by hackers. Such attacks happen all too frequently, often with unpleasant consequences. Malicious attacks can paralyze an entire organization, wipe out substantial amounts of data, and cost millions of dollars both in actual physical losses and in lost opportunity. Therefore, it is essential to have a mechanism in place that protects an organization from such eventualities.

On the other hand, organizations must be open to outside communication. You need to allow the data from your trading partners to come through; otherwise there will be no communication, let alone business integration. Furthermore, you want to make sure that communication happens quickly, and so you do not want to spend a lot of time determining whether the communication is legitimate. You must have a strategy in place to reconcile the conflicting goals of data protection and business integration before proceeding with the deployment of BizTalk Server. The following sections introduce firewall solutions and then move to deployment options related to the current firewall configuration.

Firewalls

A firewall typically is a server that stands between the private network of an organization and an outside network such as the Internet. This server forms a boundary between the organization and the outside world. It inspects all incoming and outgoing network traffic, checking all messages against specific criteria. If a message meets the criteria, it is allowed to pass through the firewall. If it

does not, it is stopped from crossing. In a way, a firewall is a network traffic filter.

Because traffic filtering requires processing time, firewalls can slow down network traffic substantially. In addition, they introduce a single point of failure in a system. If the firewall server is down, no traffic can go through, making it impossible for the organization behind the nonoperational firewall to exchange information with the outside world. Therefore, choosing the right firewall configuration is essential for the operation of the network.

Although it is possible to use just one firewall, this is not a recommended deployment. Instead, two firewalls typically are used to take advantage of the space between them. Therefore, it is recommended that an organization use two firewalls to protect BizTalk servers. Two configurations are recommended for BizTalk deployment and are discussed in the next two sections.

In both configurations it is necessary to use two firewalls and a demilitarized zone (DMZ), as shown in Figure 12.4. The firewalls typically restrict all the traffic to only HTTP and FTP, although you also might want to allow MSMQ and SMTP for some trading partners.
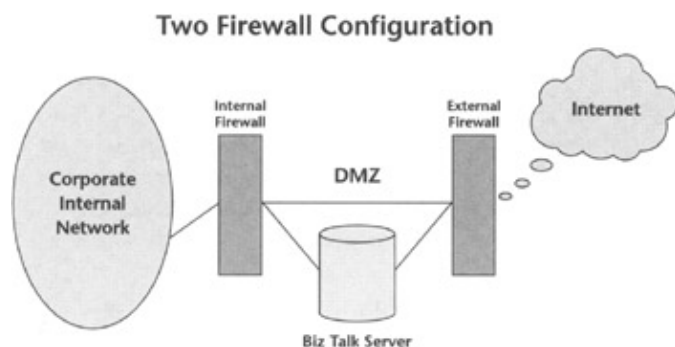

Figure 12.4: BizTalk Server usually is implemented with a two–firewall and DMZ configuration.

In both configurations the first firewall is a boundary between the Internet and the company computers. The role of the first firewall server is to inspect all the incoming traffic and apply specific criteria to each examined message. If the message meets the specified criteria, it is allowed to pass through the firewall to one of the Web servers. These Web servers are not a part of the intranet or the local area network (LAN); they are separated from the rest of the organization by another firewall. The advantage of this arrangement is that even if an attack can penetrate the first firewall, it still has one more obstacle to cross: the second firewall.

As was mentioned earlier, the area between the two firewalls is called the DMZ. Originally, a DMZ was not allowed to hold any persisted data. Only static data were allowed to cross into the DMZ. However, many business interactions, particularly those which run over substantial periods of time, require that the applications run in a dynamic context; this means that the state will change over time. For this reason, the rule about not persisting data in the DMZ had to be relaxed.

Firewall Configurations

The next two sections describe the placement of BizTalk servers in a two–firewall configuration. There are two possibilities: The BizTalk servers can be placed in the DMZ or can be placed on the internal corporate network. Each placement has some unique considerations, which are discussed next. Remember that BizTalk Server needs access to SQL Server, but it would be unwise to place the SQL server in the DMZ because such a placement would expose the data to unnecessary risks without achieving any advantages. Therefore, the SQL server is always placed behind the second firewall.

BizTalk Server in the DMZ

In this configuration you can deploy some or all of your BizTalk servers in the DMZ, but you place the SQL servers behind the second firewall, on the internal corporate network. An inbound message passes through the firewall and reaches a BizTalk server in the DMZ. If the server needs to communicate with an SQL server (which is on the corporate network), the message must pass through another firewall. The servers in the DMZ must use local transport services, that is, HTTP, Message Queue, or SMTP, only.

BizTalk Server on Corporate Networks

In this configuration BizTalk servers are deployed behind the second firewall. With this placement there are two further possibilities to choose from for the configuration: a model that allows only HTTP interactions and a message queuing fan–out model.

The choice of configuration depends on the type of transactions allowed. In the simplest case all the transactions interact by using the HTTP protocol. The result is a synchronous transaction. Synchronous transactions occur when a single message is either sent or received within the context of a single communication action.

However, not all BizTalk transactions fit the synchronous model. In some real–life situations it may be necessary to allow business transactions to last for days or weeks, and so it is necessary to plan for such long–running processes. The model that allows only HTTP interactions is not suitable for long–running processes because it does not allow for asynchronous interactions. Asynchronous transactions occur when messages are sent in pairs and occur within the context of a single communication action. The decision whether or not to allow only synchronous transactions will dictate what deployment model you use. The next two sections look at the deployment models that support synchronous transactions and asynchronous transactions, respectively.

Enterprise Application Integration

BizTalk Server can play a significant role in enterprise application integration (EAI) solutions. Many organizations have remote locations that interact with a centralized database. The remote locations have their own applications, but some of the data have to be transferred to headquarters for full analysis for the entire organization.

Currently, many of these data transfers happen over OLE DB or ODBC. While this is the fastest method, it may cause some problems as a result of driver problems and the inability to use the Internet. If both applications are on the same internal network and there are no requirements to use the Internet, OLE DB is faster than BizTalk Server and XML. BizTalk Server provides the ability to perform data transfer via XML, which can be transferred over the Internet.

For example, say you are a retail chain. You have multiple locations that have the responsibility for their own point–of–sale systems. The data are stored locally, and the new sales are entered at the store level. Currently, you are performing this with local SQL servers. The summary information is transferred to headquarters on a nightly basis. While the process is working, you are running into a couple of problems. First, each store has its own 56K connection back to headquarters. The store uses this connection for data transfer and Internet access. The connection speed is very slow. It is cheaper for the stores to get their own Internet connection with a local provider, and it would be nice to transfer data over the new Internet connection. The second problem is that some of the stores are complaining because they keep getting ODBC driver errors. They continue to have to reinstall their drivers to keep the transfer going. The solution, as shown in Figure 12.5, is to get a local Internet connection for each store and then use BizTalk Server to map the data from each store's
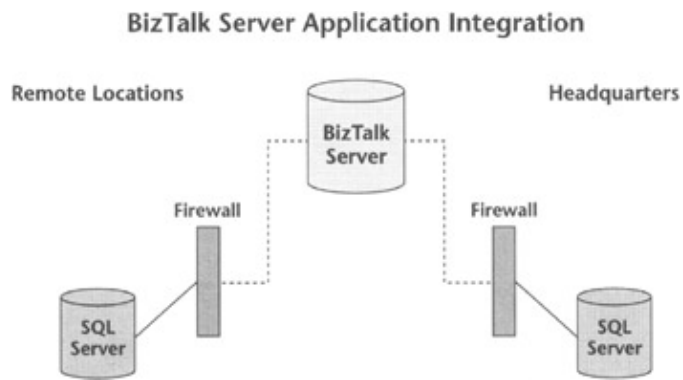
database to headquarters.

BizTalk Server Application Integration

Figure 12.5: Multiple locations can integrate business processes by using BizTalk Server.

Business–to–Business Solutions

By far the greatest strength of BizTalk Server is its ability to integrate two different organizations. Two companies can have totally different processes, applications, and databases but can be integrated with BizTalk Server. Document Specification, Transfer Maps, and Messaging Services were created just with that in mind.

For example, Org Big is a large corporation that specializes in computer equipment. You are a vendor for Org Big, selling it computer consulting services. You have an invoice–tracking system. After you perform services for Org Big, you enter your hours, and the system prints the invoice that will be mailed to Org Big. The invoice is received by the accounting department at Org Big, where the amount has to be verified before the invoice is paid. Your system is a Visual Basic .NET application that uses an SQL Server back end. Org Big's accounting system is also an SQL Server back end. The solution is to create a document specification to standardize the way your invoice is transferred directly to Org Big. Through the Messaging Services of BizTalk Server, you enter the invoice information. Instead of the invoice being mailed, the data are transferred directly to Org Big's accounting system SQL Server database. The entire process occurs via XML. This solution cuts down on mail time and decreases data entry for both organizations.

BizTalk Server can be an invaluable tool for integrating business processes, applications, and databases. The core components of BizTalk Server are based on the .NET services and enhance a company's ability to create .NET business solutions.

Review Questions

1. What is the role of BizTalk Server?
2. What are the three layers of the BizTalk architecture?
3. What is a document map?
4. Why would it be necessary to validate a document?
5. Why is it beneficial to have two firewalls for a BizTalk configuration?
6. What is the purpose of BizTalk Orchestration Services?

Chapter 13: SharePoint Portal Server

Overview

Regardless of the size of an organization, customers are looking for better, more efficient ways to share information within the organization and with outside key suppliers, partners, and clients. Microsoft SharePoint includes a set of two new technologies that facilitate information sharing both within organizations and over the Internet: SharePoint Portal Server 2001 and SharePoint Team Services. No single solution can address the information–sharing needs of an entire organization; small teams and ad hoc teams share information in ways different from those used by large teams. Consider the following:

- Small or ad hoc workgroups need informal means to work together on group deliverables, share documents, and communicate status with one another. These groups need to share information easily and effortlessly. SharePoint Team Services–based Web sites allow them to do that.
- Large workgroups with structured processes need more control over their information. They require features such as formal publishing processes and the ability to search for and aggregate content from multiple data stores and file formats. SharePoint Portal Server 2001 is recommended for this scenario.

When organizations use SharePoint Team Services and SharePoint Portal Server 2001, they can address the information–sharing challenges for both their large and their small groups. Together, the SharePoint technologies give users the ability to organize information, readily access that information, manage documents, and achieve efficient collaboration, all in a familiar, browser–based, and Microsoft Office–integrated environment.

This chapter defines the difference between SharePoint Team Services and SharePoint Portal Services and then discusses SharePoint Portal Server solutions. It concludes with a section on interaction with the other .NET Enterprise Servers. The chapter focuses on the relationship between Exchange Server and SharePoint Portal Server, but the relationship of SharePoint with Content Management Server also is referenced.

SharePoint Technologies

The SharePoint technologies allow users to share data more efficiently. This section discusses the differences between SharePoint Team Services and SharePoint Portal Server. After reading the following sections, combined with Table 13.1, the reader should be able to determine the SharePoint technology that is appropriate for his or her organization. Table 13.1 compares and contrasts the SharePoint technologies.

Table 13.1: SharePoint Technologies

CATEGORY	TEAM SERVICES	PORTAL SERVER
Core function	Ad hoc information sharing	Enterprise search and sharing
Web site	Team Web sites	Portal Web sites
Storage	SQL Server	Exchange Server Web Storage System

Document management publishing	Check in, check out, versioning, routing, publishing	
Client applications	Browser, Office XP, FrontPage 2002	Browser, Windows Explorer, Office XP and 2000

SharePoint Team Services

SharePoint Team Services gives users the ability to quickly create and contribute to team– or project–focused Web sites from their browsers or Office XP applications. With SharePoint Team Services, teams quickly can create a Web site for sharing information such as documents, calendars, announcements, and other postings. Also, Web sites created with SharePoint Team Services are easy to customize and manage even for those who have never created a Web site. SharePoint Team Services is included with the FrontPage 2002 Web site creation and management tool and the versions of Office XP that contain FrontPage. Microsoft plans to include the technology in upcoming releases of the Windows Server operating system and other products.

Microsoft SharePoint Portal Server 2001

SharePoint Portal Server 2001 creates a portal Web site that allows users to share documents and search for information across the organization and enterprise, including SharePoint Team Services–based Web sites, all within one extensible portal interface. SharePoint Portal Server includes robust document management features that allow companies to incorporate business processes into their portal solutions. SharePoint Portal Server is a standalone server product that was released in the first half of 2001.

SharePoint Solutions

Small workgroups often use a combination of email, file servers, and their own hard drives to store and share information. This type of information sharing has become the status quo for team information sharing because all the members can participate easily and equally, but it does not create an organized record of a team's efforts. Using a SharePoint Team Services–based Web site gives teams an easy and informal way to centralize and share project and team information with team members and other interested parties within an organization.

At an organization– or business–division level, however, information–sharing requirements become more sophisticated. By using SharePoint Portal Server 2001, an organization can aggregate content from across the organization into a portal Web site so that the users can find the information they need to make better business decisions regardless of where the data reside. This requires comprehensive search capabilities and the ability to manage large volumes of information across a great number of data stores. Business units also need advanced document management, including structured publishing processes, to ensure that the information they are sharing is complete and up to date.

As shown in Figure 13.1, a user interacts with a SharePoint Portal Server workspace in three primary ways:

- Through the SharePoint Portal Server dashboard site
- By using the client extensions of SharePoint Portal Server for Windows Explorer and Web folders
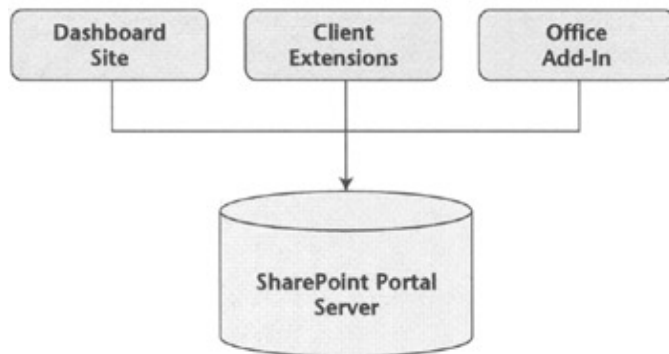- By using the SharePoint Portal Server Office add–in

Figure 13.1: A user can interact with a SharePoint workspace through any of three different access methods.

The following sections outline the types of solutions that can be provided by the SharePoint technologies. The reader is introduced to the Digital Dashboard Services and then provided with an outline of the client extensions. This followed by a brief introduction to the Office add–in. After these solutions are presented, there is a description of the security model. Managing the security model is the key to making the SharePoint technologies work for an organization.

Digital Dashboard Services for the SharePoint Portal Server

The SharePoint Portal Server dashboard site uses Microsoft Digital Dashboard technology to create a modular, configurable, and extensible user interface for SharePoint Portal Server workspaces. Clients using Web browsers such as Microsoft Internet Explorer can access the document management commands, search, announcements, and other SharePoint Portal Server features by using the digital dashboard.

> **Note** The SharePoint Portal Server Web user interface is referred to as the SharePoint Portal Server dashboard site. This site is the premier Web storage system implementation of the Microsoft Digital Dashboard architecture. Each page in the dashboard site is a subdashboard composed of one or more Web parts. Each Web part implements a basic workspace function, such as running a search query, displaying the search results, browsing the hierarchy of document folders, or displaying the list of documents in a folder. A Web part either implements the workspace function by using the Document Management Object Model (PKMCDO), Collaborative Data Objects (CDO), or ActiveX Data Objects (ADO) object models or calls a back–end service such as those exposed by the Microsoft Document Management or Microsoft Search process. More information about these types of object calls appears in Chapter 7, "Exchange Server."

Digital dashboards are applications that run on the Windows 2000 Internet Information Services platform. SharePoint Portal Server digital dashboard Web parts are stored in the Web storage system as part of the workspace. The coordinator can customize the portal to suit the needs of the business and its users. The coordinator can add custom Web parts to the portal and create new workspace dashboards.

By using the Microsoft Digital Dashboard framework and the user interface for portal configuration, programmers or coordinators can modify Web part settings and add new dashboards and Web parts.

A couple of shared user interface elements are common to the three ways in which users can access the SharePoint Portal Server document library:

- SharePoint Portal Server Document Properties dialog
- SharePoint Portal Server Document Edit Profile dialog

The Document Properties dialog displays the following types of information:

- Basic document property
- Profile information
- Version history
- Security settings
- Category information

If the document is an Office document, the Web storage system automatically promotes the Office document properties to Web storage system item properties.

In the Properties dialogue for the document profile, the user can update the version comments, select a new profile template, and provide the values for the template's profile properties. All these profile properties are stored as item properties in the Web storage system, where they can be searched and accessed with ADO.NET. The Web storage system used by SharePoint Portal Server is the same as the Web storage system available from Exchange Server 2000. An organization can have a SharePoint Portal Server use its storage system or the one provided with Exchange Server. SharePoint Portal Server can be integrated with Exchange Server 2000 to build on the current public folder system of Exchange Server 2000. There is more information on the Web storage system in Chapter 7, "Exchange Server."

Client Extensions

A user also can access a workspace by using Windows Explorer (together with Windows Web folders and the client extensions of SharePoint Portal Server). Additional administration functions are available through the Windows Explorer Web folders user interface. Windows Explorer Web folders provide direct access to the workspace's support and administration folders.

These client extensions allow a user to open Windows Explorer and have direct access to a workspace. This is an easy method for sharing and accessing data. Most users are comfortable with Windows Explorer, and this functionality is appreciated.

Microsoft Office Integration

The SharePoint Portal Server Component Object Model (COM) add−in for Microsoft Office integrates the SharePoint Portal Server document management functions as additional menu items in Word, Excel, and PowerPoint. SharePoint Portal Server supports both the Microsoft FrontPage WEC Extensions and Office Server Extensions protocols. The Office COM add−in integration uses the Web storage system's native WebDAV to support SharePoint Portal Server document management functions from the Office applications.

SharePoint Security

Security is essential for both document management tasks and the search function. In document management it is important to restrict access to sensitive information. In document approval scenarios it is important to restrict the viewing of a document to those who can edit or approve it

until it is ready for a larger audience. For search it is important that SharePoint Portal Server recognize security settings for crawled content so that users viewing the results of searches cannot see documents to which they have no right to access.

SharePoint Portal Server recognizes any security policies currently assigned to an organization's servers, file shares, and databases. For example, when SharePoint Portal Server crawl documents are stored on an organization's servers, the security policy on each document is upheld when SharePoint Portal Server provides search results.

The following sections provide a more complete definition of the security architecture of SharePoint Portal Server. The reader is introduced to role–based security and then to the resulting security access when users belong to more than one role. Finally, there is a discussion of managing and administering the roles.

Role–Based Security Architecture

Assigning a role to a user gives that user permission to perform specific tasks. For example, a user assigned to the SharePoint Portal Server author role has permission to add new documents to a folder, edit all the documents in the folder, delete any document from the folder, and read all the documents in the folder.

SharePoint Portal Server uses a fixed set of three roles to offer a secure method for controlling user access to workspace documents. It is not possible to modify role permissions. Although roles can be set at the workspace level, they usually are set at the folder level. In addition, it is possible to completely deny a user or users access to a specific document.

The SharePoint Portal Server role–based security architecture includes the following roles, as described below:

- Reader
- Author
- Coordinator

A *reader* can search for and read documents but cannot add them to the workspace. All folder users have reader permissions by default. In an enhanced folder readers can view only folders and published versions of documents. A reader cannot check out, edit, or delete workspace documents and cannot view draft document versions. When the workspace is created, SharePoint Portal Server assigns the Windows 2000 Everyone group to the reader role for all the folders in the workspace.

An *author* can add new documents to a folder, edit all the documents in the folder, delete any document from the folder, and read all the documents in the folder. In an enhanced folder authors also can submit a document for publishing. An author can create, rename, and delete folders. When a new folder is created, the roles and folder policies are inherited from the parent folder. However, the author cannot change the roles or the approval policy on folders he or she creates.

A *coordinator* on the workspace node manages content in the top–level folder and performs a set of workspace administration tasks. These tasks include managing content sources, document profiles, categories, and subscriptions and customizing the portal. The coordinator creates indexes of updated content when necessary or schedules this to occur automatically. SharePoint Portal Server automatically assigns the administrator who creates the workspace to the coordinator role on the workspace node and on each of its folders. A coordinator on a specific folder configures user roles

for that folder. The coordinator creates subfolders as well as adds, edits, and deletes documents from them. Coordinators also can read and delete a document that has been created but has not yet been checked in. For enhanced folders the coordinator selects the appropriate approval process. In addition, the coordinator can undo the checkout of a document or end the publishing process by using the Cancel Publishing or Approve Now action.

SharePoint Portal Server provides the Deny Access security option on documents only. This setting supersedes all access permissions except those of the local administrators group. It is possible to deny access to a document for a specific user or group if the organization does not want that user or group to view that document. Denying access to a document does not affect the local administrators group's access to that document.

Additionally, SharePoint Portal Server includes a set of folders to support workspace management functions: the Management, Portal, System, Shadow, and Categories folders and their subfolders. A user must be assigned to the coordinator role on the workspace node to manage these folders. Security cannot be specified directly on these folders, and the folders generally are not exposed to end users.

The Windows 2000 local administrators group has permission to read documents and configure security on any folder or document in a workspace. The ability to configure security provides a way to access every folder in case a folder is made unavailable to those who should have access to it.

The local administrators group can restore permissions for individual folders. Denying access to a document does not affect the local administrators group's access to that document.

Users Can Have More Than One Role

A user can have different roles for different folders in the same workspace. For example, in one folder a user may have reader permissions only, while in another folder the same user may have author permissions.

A developer can give groups of users access to folders in the workspace as though they were a single user by assigning the group to a role, such as reader. If an individual user is assigned to more than one role in a folder (as a member of a group and as an individual), the most permissive combination of rights takes precedence. However, a developer also can deny a user or group access to a specific document, and this supersedes all other permissions associated with roles. Because access to a particular document can be denied, a user can have one role on a folder but have no access to a document within that folder.

Role Administration

An administrator on the SharePoint Portal Server computer can assign users to roles on the workspace node by using the SharePoint Portal Server console in the Microsoft Management Console (MMC). In addition, SharePoint Portal Server automatically assigns the administrator who creates the workspace to the coordinator role on the workspace node and on each folder.

The Windows 2000 local administrators group has permission to read documents and configure security on any folder or document in a workspace. Denying access to a document does not affect the local administrators group's access to that document.

A SharePoint Portal Server computer administrator manages roles on the workspace node by using SharePoint Portal Server console in MMC on the SharePoint Portal Server computer. If you have

coordinator permissions on the SharePoint Portal Server workspace node, you can assign users to roles on the workspace node and on any individual workspace folder that inherits its security setting from that node. In addition, a user must be assigned to the coordinator role on the workspace node to manage the set of folders that support workspace management functions. The security settings also can be managed in each workspace's management Web folder.

A coordinator manages roles at the folder and document levels by using the folder or document properties pages in the workspace. To do so, the coordinator must have the client components of SharePoint Portal Server installed on his or her computer with Windows 2000. Microsoft Windows NT and Windows NT Version 4.0 users cannot manage SharePoint Portal Server roles.

.NET Applications with SharePoint Portal Server

SharePoint Portal Server provides a medium for the development of collaborative applications. Collaborative applications can be built three ways using SharePoint Portal Server:

- Within an existing SharePoint Portal Server workspace.
- Within a non–SharePoint Portal Server public folder on a SharePoint Portal Server computer (for example, outside the SharePoint Portal Server applications and workspaces folder hierarchies). This option integrates Exchange Server with SharePoint Portal Server.
- As a non–SharePoint Portal Server, non–Web storage system application that needs to integrate programmatically with an existing SharePoint Portal Server or non–SharePoint Portal Server Web storage system application.

Figure 13.2 depicts three ways a user can access and interact with a SharePoint Portal Server workspace (in particular, the enhanced Documents folder): the SharePoint Portal Server dashboard site, the client extensions of SharePoint Portal Server for Windows Explorer Web folders, and the Office COM add–in.
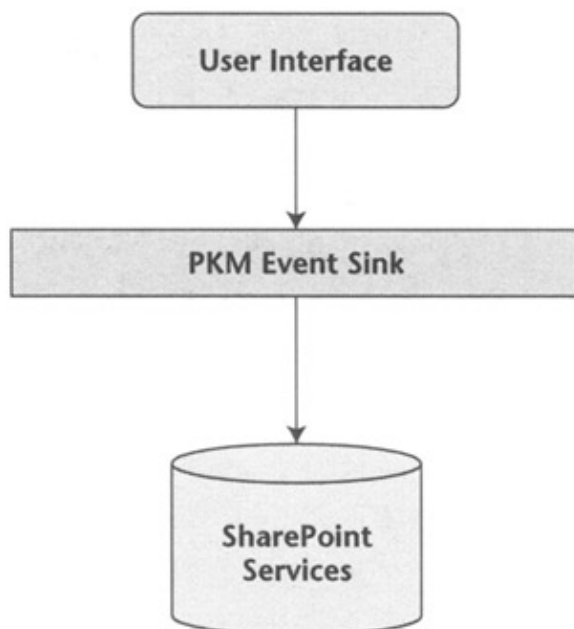

189

Figure 13.2: Every request that goes to SharePoint Portal Server must go through the PKM event sink. It is through the PKM event sink that applications can invoke certain features of SharePoint Portal Server, such as document approval routing and drag–and–drop functionality.

As shown in Figure 13.2, the PKM event is a back–end database store level event sink. An event sink is an action that is associated with an event. Event sinks can be used to customize user–related features. The information store process manages the folder hierarchy and contents (folder items) in the Web storage system. The SharePoint Portal Server Document Management and Microsoft Search processes provide their respective services to the client components of SharePoint Portal Server.

The following sections describe the options for user interaction with SharePoint Portal Server in more detail.

Workspaces

Aside from the coordinator's ability to customize the content and layout of the SharePoint Portal Server dashboard site by using the digital dashboards and Web parts that accompany SharePoint Portal Server, developers can provide additional customization by developing their own Web parts. Creating a new, customized function in SharePoint Portal Server requires consideration of the following questions:

- What changes, if any, are required to the interface from the perspective of a SharePoint Portal Server dashboard site user to deliver end–user functionality?
- Which SharePoint Portal Server services support the new functionality?
- What are the SharePoint Portal Server folder requirements or other information storage requirements that are needed to deliver the new functionality?

Implementing customized functionality usually requires creating a new Web part that then calls the SharePoint Portal Server back–end services to provide the required results. In addition, a Web part can call any service running on that server or another server or do this through custom Microsoft .NET Web Services running on a Web site across the Internet. Although it is possible to reuse and extend some of the Web parts that accompany SharePoint Portal Server, this must be done carefully.

When the new functionality uses SharePoint Portal Server folders (folders in the SharePoint Portal Server applications or workspaces folder hierarchies), it is critical that the new functionality not configure any of its own store–level event sinks or workflow designer event logic. Solutions that require this type of functionality must be implemented outside the SharePoint Portal Server workspace hierarchy.

This also applies to reusing the SharePoint Portal Server schema (also known as content classes). It is recommended that the new solutions use their own solution–specific content classes derived from the underlying Web storage system or SharePoint Portal Server content classes where possible.

An example of this type of enhanced functionality is a new version of the announcement Web part that allows users to submit their own announcement items and have the submission process managed by the SharePoint Portal Server document management checkin, approval routing, and publish functions. Implementing this Web part requires the creation of a new Web part (perhaps modeled after the existing announcement Web part). The role of the Web part would display only the announcement items that have been approved for publishing. In addition, the Web part could link to a Web storage system form that would allow users to create new or edit announcement items

easily.

Alternatively, users could be asked to create their announcements by using Word and save them by using a custom document profile. The document profile would capture the properties that are important to announcement items. From a storage perspective, these items would have to live in their own enhanced document management subfolder (separate from the default announcement folder created when a user creates a new workspace). Document approval routing would be enabled to ensure that new announcements were approved before they appeared in the new Web part. The Web part would display only those announcements whose status has been *published* (meaning they have been approved).

Collaborative Solutions with Public Folders

In the same way that collaborative solutions can be built on the Web storage system used by Exchange 2000 Server, applications can be built by using the Web storage system of Exchange Server and use the functionality provided by the SharePoint technologies.

The Web storage system includes the forms registry and the forms renderer. The forms renderer creates the user interface. Alternatively, the non−SharePoint Portal Server public folder application could be displayed as a Web part. In this case the Web part definition lives under the portal subfolder of the workspaces where it is used. The content and event sink registrations reside in a non−SharePoint Portal Server public folder. This can be a public folder within Exchange Server.

A Web part can make use of any database, directory service, or SharePoint Portal Server service. The databases most commonly used with SharePoint Portal Server are Exchange Server public folder stores. They could, however, be SQL Server databases. In addition, a Web part can use any other application service running on the SharePoint Portal Server computer or another server within the domain. Web parts also can draw upon any custom XML Web Service running on a Web site across the Internet.

The SharePoint Portal Server role−based security model is enforceable only when the underlying Web storage system is accessed through the ADO and CDO object models or through the WebDAV Internet protocol. WebDAV is an extension to HTTP that became available with HTTP Version 1.1. WebDAV allows posting to a Web site or Web page via HTTP over the standard port 80 protocol. Accessing the Web storage system by using the Installable File System (IFS) bypasses the SharePoint Portal Server role−based security mechanisms. The IFS driver typically presents the hierarchical folder store as a hierarchical file mounted as the M: drive. When Exchange Server installs the IFS, the exchange databases are made available through a mapping of the M: drive. If you have the appropriate permissions, you can navigate through the data as though it were any other file on the operating system. Therefore, the server must be secured with respect to local physical access as well as network and terminal services access. For more information on IFS, refer to Chapter 7, "Exchange Server."

Collaborative Solutions with External Applications

Collaborative solutions also can be built when a new or existing non−SharePoint Portal Server, non−Web storage system application wants to use the SharePoint Portal Server document management or index and search functions. This common scenario is easy to design and implement by using any of the available object models or Internet protocols: PKMCDO, CDO, ADO.NET, and WebDAV.

PKMCDO reveals virtually all the SharePoint Portal Server document management functions, including the ability to create standard and enhanced folders, check in, check out, and publish documents.

PKMCDO can add a tremendous amount of functionality to applications. For example, say your organization has a human resources package that stores company information about new job postings. Through the human resources package you can run reports to see the information, and the results are displayed in a Word document. You could, through PKMCDO, use SharePoint Portal Server to take the document from the human resources application and publish it on a corporate intranet site. The IFilter object model within PKMCDO allows the indexer to read and parse any document format. Because you can store all types of documents in a SharePoint Workspace, the index service needs to know how to read different types of files. The IFilter object provides indexing features with the functionality to open all recognized file types. When the files are open, the indexing service can read the files and parse the words to be stored in an index that is similar to the index in the back of a book. When a user searches for a word, the index is searched and the hits are displayed with pointers to the pages that meet the search request.

Summary

SharePoint Portal Server is a Web portal and a new platform for document management and enterprise indexing, searching, and collaboration. SharePoint Portal Server enables developers with basic or advanced skills to create collaborative end–user solutions easily and quickly. It can be integrated with the Web storage system and IFS of Exchange Server to create solutions that are integrated with the current .NET solutions.

SharePoint Portal Server uses a role–based security model that allows you to control the access to pages at the portal. The organization can use SharePoint Portal Server to bring Internet publishing solutions to the user's desktop. Users can access the portal through the following interfaces:

- The SharePoint Portal Server dashboard site
- The client extensions of SharePoint Portal Server for Windows Explorer and Web folders
- The SharePoint Portal Server Office add–in

The role–based security model allows you to control the user interaction. One of the primary concerns about bringing a Web portal closer to the user is security. You should be comfortable with the ability of the security model of SharePoint Portal Server to control the publishing process.

Of particular value to collaborative–solution developers is the Web storage system, with or without Exchange Server 2000, on which SharePoint Portal Server is built. The Web storage system provides a flexible platform for building new document–based, workflow–enabled collaborative Web solutions. You can create collaborative solutions that are dependent on SharePoint Workspaces and public folders or are integrated with external applications. A developer has great flexibility for collaborative solutions when working with the SharePoint technologies.

Review Questions

1. What are the three options for accessing SharePoint Portal Server?
2. What are the two SharePoint technologies? What is the difference between them?
3. What is the PKM event sink?

4. What three types of .NET applications can be created with SharePoint Portal Server?
5. What is role–based security?

Chapter 14: Internet Security and Acceleration Server

Overview

Microsoft developed the Internet Security and Acceleration (ISA) Server, an extensible enterprise–class firewall and Web cache server, as an upgrade from Proxy Server 2.0. It is designed to work with and take advantage of the functionality in the .NET Server operating system as well as Active Directory. ISA Server has a multilayer firewall that protects network resources from outside intrusions such as viruses, hackers, and unauthorized access. It increases an organization's ability to provide fast Internet access by building a Web cache that retrieves Web objects locally rather than through the Internet. Added functionality with the ISA Server allows a user to have a firewall, a Web cache server, or both if he or she chooses. ISA Server not only provides good functionality, it includes sound management tools that enable easy and quick access to centrally located management and administrative tasks. Through the Microsoft Management Console it is possible to incorporate these tools with other management snap–ins that can be used on a regular basis.

Often the firewall seems to be a deterrent to effective application development. As an organization develops enterprise applications, it should be comfortable with the configuration of the ISA Server. This makes configuration of the application easier. The firewall configuration of ISA Server can change connection options and in some cases help a user determine the protocol and data formats that should be used for application data flow. ISA Server also has some features that are implemented to help with Enterprise development. Extensible Markup Language (XML) Web filters can be created to filter Web content on the basis of any criteria the organization chooses. The developer can look at the details of the request, not just the source Internet Protocol (IP) address, port number, or protocol.

ISA Server has taken on a larger role within organizations as the other .NET Enterprise Servers have become more popular. Over the last several years many organizations have struggled to make many of their applications available on the Internet. One of the primary concerns related to Internet deployment is security. The data have to be available, but most organizations are concerned about who has access to the data and which servers have to be directly accessible from the Internet. In many cases the applications are deployed with a Web server, which also may be running Application Center 2000 or Commerce Server, sitting outside the firewall (ISA Server) and the data residing in a database server (SQL Server) residing behind the firewall. The data are protected by the ISA Server. Web users are allowed access to the Web server, and the Web server then makes the connection back to the data through the firewall.

Additionally, ISA provides an opportunity to take advantage of Server Publishing. Server Publishing makes it possible to deploy an application with the Web server and the database server residing behind the firewall. The Web user interacts with ISA Server, which communicates with both the Web server and the database server on the user's behalf. The Web user never has direct access to the servers that are providing the application.

This chapter begins by introducing the available ISA Server editions. It then identifies the modes that are available with ISA Server. Firewall mode is described in detail in this section because it is crucial to the deployment of enterprise applications to the Web. Next, the role of ISA Server with the .NET Enterprise Servers is described. Finally, this chapter breaks down the interoperability with the other .NET Enterprise Servers.

Comparison of Editions

The ISA Server is available in two editions: the Standard Edition and the Enterprise Edition. The next few paragraphs cover the differences between these editions. The discussion and features described in this chapter almost exclusively cover the Enterprise Edition. The features required for Enterprise development are amplified with the Enterprise Edition.

Standard Edition

The Standard Edition can be thought of as a standalone ISA Server. It is on its own, separate from the directory structure of the organization's domain. This ISA Server does not need to belong to a domain. All the policies for determining incoming and outgoing connection strategies are defined and maintained on this server.

For small offices/home offices (SOHOs) and small to medium−size businesses this is a sound option. It is less expensive and does not require the organization to have deployed Active Directory. The Standard Edition includes the following identifying features. Each of these features is described in more detail later in this chapter:

- Enterprise class firewall
- Web caching of external Web requests
- Support for up to four processors
- Server and Web publishing of internal resources

Although the product provides many of the core features, it does have some limitations for larger organizations. It cannot scale past four processors, integrate with Active Directory, or take advantage of caching strategies across multiple servers. The Standard Edition is on its own and does not integrate well with an existing directory structure or satisfy the need to scale to more ISA Servers. Table 14.1 identifies the core differences between the two editions of ISA Server.

Table 14.1: ISA Server Editions

FEATURE	ISA SERVER STANDARD EDITION	ISA SERVER ENTERPRISE EDITION
Server deployment	Standalone only	Multiserver with centralized management
Policy level support	Local policies only	Enterprise and array−level policies
Hardware scalability limit	Four processors	Limited only by the operating system
Caching	Cache is stored locally	CARP can be used to distribute cache across servers

Enterprise Edition

The Enterprise Edition overcomes many of the the difficiencies of the Standard Edition. The Enterprise Edition was designed to meet the performance, management, and scalability needs of high−volume Internet traffic environments. With the Enterprise Edition a user can take advantage of several features that are not available in the Standard Edition. Remember, with the Enterprise Edition the purchaser gets all the features of the Standard Edition; he or she also gets a higher price tag:

- The user is limited in the number of processors only by the version of the .NET Server platform that has been installed.
- Integration with Active Directory allows for security policies that are tied to the same account that already is managed in Active Directory. This is among the primary enhancements for enterprise development. Through Active Directory, the ISA Server can be tightly integrated with the .NET Server platform.
- Multiple servers can be managed as a single logical unit.
- Clustering and fault tolerance are available through the implementation of arrays.
- Enterprise and array–level policies are used to apply a standard configuration to multiple servers.
- Cache Array Routing Protocol (CARP) can be used to destribute the cached Web site responses across multiple servers.

Decision

The decision between the two editions depends on the features the user needs compared with the expense of the product. Many of the core features related to security, caching, management, extensibility, and even performance to some degree are the same in both editions. What sets the two editions apart from each other is the scalability and fault–tolerant solution that is available with the Enterprise Edition. Table 14.1 describes the differences between the two editions.

ISA Configurations

ISA Server can be installed in three different modes, and the chosen mode can be configured during installation. It also can be adjusted at a later time.

First, ISA Server can be installed in Firewall mode. In this mode it acts as a firewall to the outside world. It can be configured to block or allow certain types of traffic to and from the Internet. This mode is critical to the secure deployment of enterprise applications. In many cases, a Web–based application is deployed with the Web server sitting outside the firewall and the database server sitting behind the firewall. The Web user connects to the Web server, which communicates with the database server. The firewall has to be configured to allow the Web server to communicate effectively with the database server. This involves opening the correct ports on the firewall to allow for connectivity. Firewall mode is described in more detail in the next section.

Second, ISA Server can be installed in Cache mode. Cache mode provides no firewall protection; however, it can be used to increase the performance of clients to as well as from the Internet. Cache mode is not typically a concern for Web–based enterprise development.

Third, ISA Server can be installed in Integrated mode, which is the most common configuration. This mode provides a firewall as well as a caching server, and this is the best use of both modes incorporated into one ISA solution. ISA Server can be quite flexible in meeting a user's needs for enterprise development while speeding up access to Internet resources. The modes of configuration for ISA Server are described in the next couple of sections.

| Note | If a user alters the configuration mode after installation, it is necessary to restart the ISA Server services for the change to take effect. This results in a short downtime and should be scheduled appropriately. |

Firewall Mode

Implementing ISA Server as a dedicated firewall means that its sole concern is what is coming in and going out or, more specifically, what is going through it. This makes ISA Server transparent to internal and external users. As information passes through the ISA Server, the server verifies that what is passing through is allowed or is blocked. This also depends on what rules have been defined for the firewall. The ISA Server will act as a gateway to the Internet for the organization. Although the ISA Server acts as an effective deterrent for internal users from sites or content they should not access, most organizations want to use the firewall to protect internal resources from Internet users.

Protecting internal resources has become more complicated as the concept of enterprise development has grown to include the Internet as a medium. Internet accessibility is the purpose of several of the .NET Enterprise Servers, including Application Center 2000 and Commerce Server. Application development often includes a tiered model that includes the Internet. The most common strategy for Web–enabling an application is to place the Web server outside the firewall and place the database server behind the firewall. In this configuration the Web user accesses the Web server, generally via port 80. The Web server makes a connection to the database server, generally via the database server port (1433 is the default port for SQL Server) or over port 80, using XML for data retrieval. The advantage of this model is that the Internet client never has direct access to the database server. Additionally, the ISA Server, which is filtering the packets, has to allow access to the database server only from the Web server. Even though port 1433 is open on the firewall, it should be open only to the IP address of the Web server. This prevents unwanted Internet users from connecting to port 1433 through the firewall. The concept of opening a port to only a single machine (IP address) often is referred to as a tunnel. ISA Server allows a user to publish an application to the Internet securely.

During the installation of ISA Server the user is asked for the configuration mode, as shown in Figure 14.1.
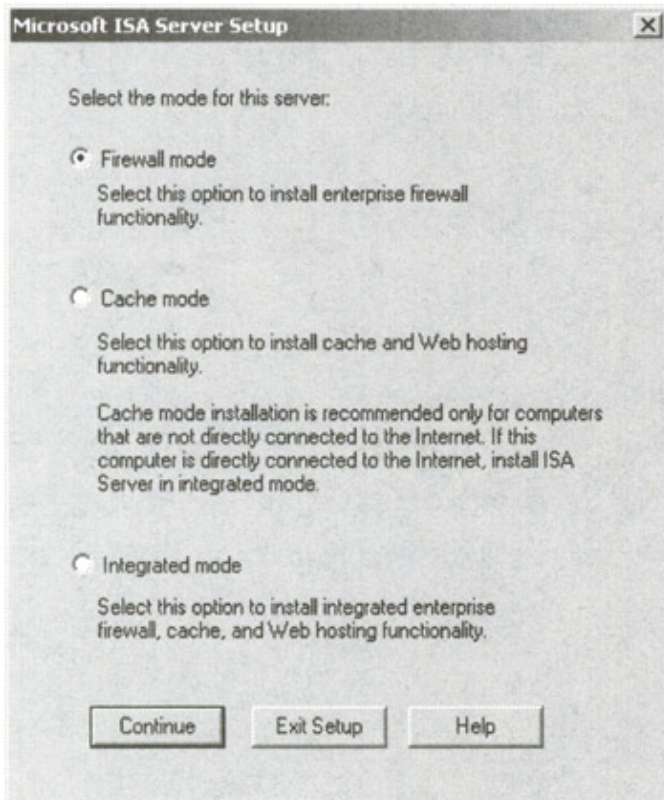
197

Figure 14.1: During the installation the user chooses the appropriate configuration for ISA Server mode.

If you would like to change your server configuration after installation, you need to perform a couple of tasks. First, you have to enable the server or array for packet filtering. To accomplish this, perform the following steps:

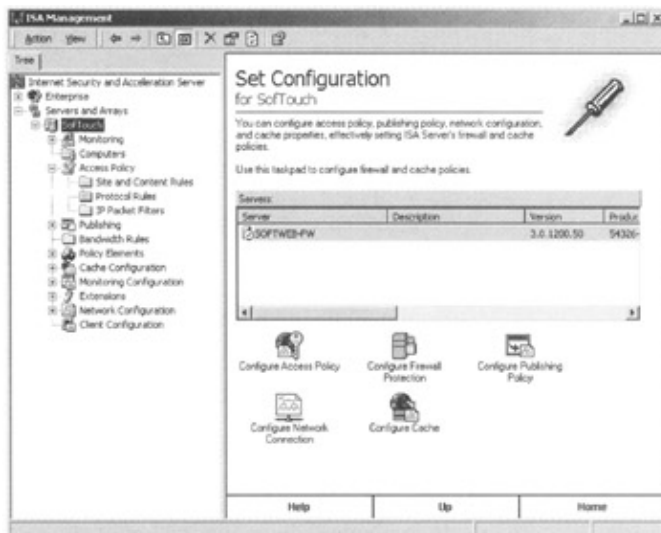1. Open ISA Management from the Microsoft ISA Server program group, as shown in Figure 14.2.

Figure 14.2: The ISA Management console tool performs the configuration for an organization's ISA Server.

2. Click to expand Server and Arrays.
3. Right–click the array you would like to alter and select Properties.
4. Click the Policies tab.

5. Click the checkbox to Force Packet Filtering on the array, as shown in Figure 14.3.
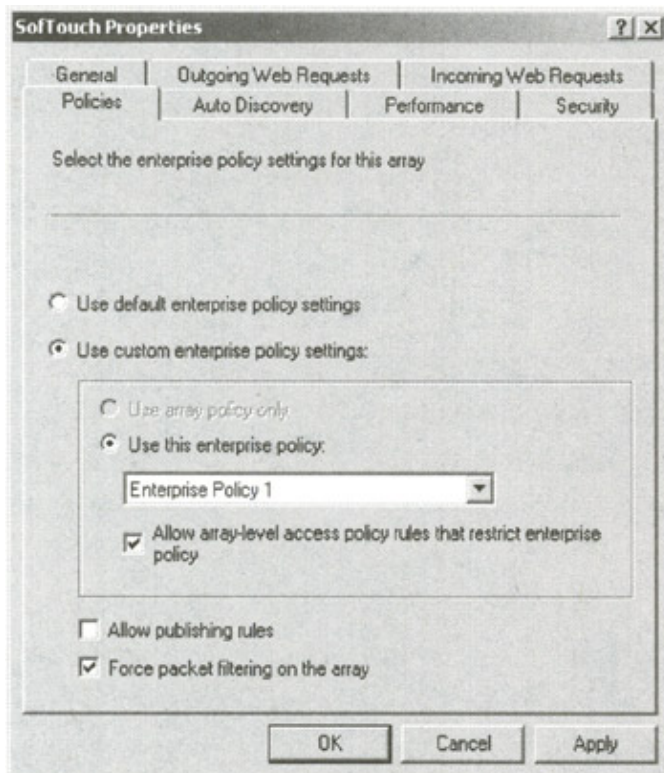
Figure 14.3: Packet Filtering should be configured at the array or server level to configure a server for Firewall mode.

After you have configured the server or array for packet filtering, you must make sure that packet filtering is enabled for the server. If the server is not configured for packet filtering, it cannot inspect the packets that come in and go out of it. To configure the server with the ability to filter packets, perform the following steps:

1. Open ISA Management from the Microsoft ISA Server program group.
2. Click to expand Servers and Arrays.
3. Click to expand the server or array the user wants to configure.
4. Click to expand Access Policy.
5. Right–click Packet Filters and select Properties.
6. From the IP Packet Filters Properties dialogue box, as shown in Figure 14.4, select Enable Packet Filtering.
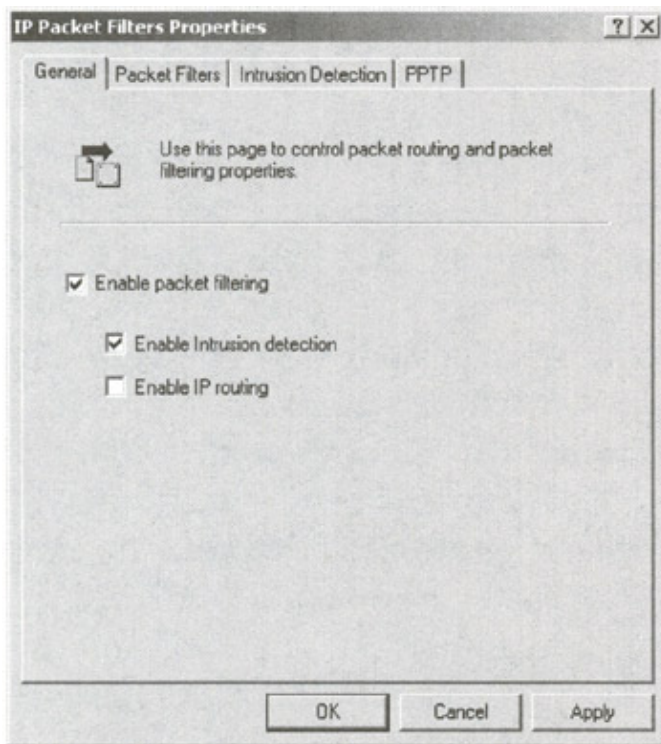
Figure 14.4: Each server can be configured to filter the packets that come in and go out of its external interface.

Cache Mode

When you choose to implement your server in Cache mode, you are not concerned about the firewall features. This may be the case if you have chosen another product to perform the needed firewall services. At this point you are configuring the server to control the access of internal users to the Internet. The caching features can be used to speed up access to the Internet for internal users and proxy their requests to the Internet. The caching features work by requesting an object from the Internet on behalf of a client. After ISA Server retrieves the object, it stores the object in the cache to return when other requests for the object are made. This allows many Web requests to be resolved locally without the need to connect to the Web server over the Internet. The proxy feature allows internal clients to use private IP addresses and have them translated by ISA Server to a valid Internet IP address.

In a Cache mode the firewall features are not enabled. The packet filtering options described in the section titled _Firewall Mode_ must not be enabled. While this helps speed up access to the Internet, it does not protect published applications. This mode is common among organizations that are using another product for the firewall solution and want ISA Server only to accelerate Internet requests.

Integrated Configuration

Its caching features make ISA a sound solution for Internet connectivity. Organizations are concerned about security. Of course, cost is an important factor. ISA allows you to combine a firewall solution with a Web solution that is sound and can be cheap compared with some of the many alternatives. Being able to integrate a firewall with a Web caching server is a big plus. If you have chosen one mode over the other during installation and would like to take advantage of features related to both modes of configuration, you have to make sure that caching is enabled and that the firewall features are enabled. After you have done this, you will have to restart the ISA services. You then are in Integrated mode.

Role of ISA Server with .NET Services

ISA Server represents a significant upgrade from Proxy Server 2.0. While many of the features are the same or similar, they have been enhanced siginifically. Here, the primary focus is the role of ISA Server with the other .NET Enterprise Servers and .NET services. For more information about each of the ISA Server features, refer to the ISA Server product documentation or the ISA Server homepage at http://www.microsoft.com/. This section breaks down the features that are integrated with or facilitate .NET services. It outlines the features related to packet filtering (extended by XML Web filters), introduces Server Publishing as an alternative strategy, and finally covers the enterprise policies of ISA Server.

Packet Filtering

Packet filtering is critical to any firewall. It is the concept of analyzing the contents of a network packet to determine whether they should be allowed to pass. This concept is the core ideology of any firewall strategy. ISA Server provides several options for filtering packets. A packet can be filtered by protocol, port number, or IP address or extended through the use of XML to filter XML traffic that typically passes over port 80.

Packet filtering is used to prevent unwanted traffic from entering an internal network. Although this is a security benefit, packet filtering is more intrusive than routing the packet and slows down the transmittal of data. This is necessary in most cases, but if you set up several layers of firewalls and filter packets in multiple places, you will increase the overhead multiple times and could slow application performance considerably.

Packet filtering is critical to Web–based enterprise development. This is the most common scenario for the deployment of Web–based applications. The ISA Server and a router are used to create a demilitarized zone (DMZ) where the Web server is located. This Web server is accessible to Web users. The Web server stores the Web application but typically does not store the data. The application includes code that connects to a database server that sits behind the firewall. When the user requests data from the application, the request is passed from the Web server through the ISA Server that is acting as the firewall and on to the database server. The following example lays out a common configuration.

The organization has an application for data analysis. The application is written in ASP.NET and is installed on an Internet Information Server (IIS). The ASP.NET code connects to an SQL Server database to retrieve the data that are used for the analysis. The following application could be configured as shown in Figure 14.5.
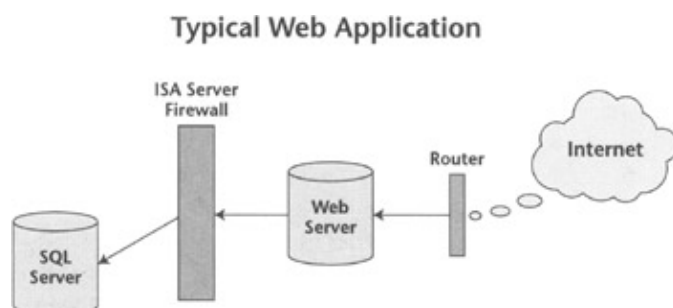

Figure 14.5: A Web application commonly publishes the application outside the ISA Server, with the data behind the ISA Server.

The Internet connection comes into the network and connects to a router. The router has a connection to a switch. One port on the switch is connected to an ISA Server that acts as the firewall. The ISA Server has two network cards. The first is connected to the switch that goes out to the Internet, and the second is connected to the internal network. All internal users have to go through the ISA Server to get to the Internet, and all Internet users have to go through the ISA Server to get to the internal network. The other ports on the switch that connect the ISA Server and the router are used to connect servers that are connected directly to the Internet. These machines also are accessible from the Internet. This area is referred to as the DMZ.

The Web server and Web application are installed on a server that is connected to a port on the switch in the DMZ. Web users connect to the Web server by using standard port 80. The Web server then connects to an SQL Server that is behind the ISA Server on the internal network. For the connection to occur, the request has to pass through the ISA Server. This connection is performed over port 1433.

With this configuration the firewall does not allow port 80 traffic to go through to the internal network; it allows only port 1433 traffic to enter the internal network. To allay the security concern of others connecting to the SQL Server, the connection is allowed only from the IP address of the Web server. This configuration provides a securely published Web–based application.

This example could be expanded as the application grows. If you had too much traffic for a single Web server to handle, you could add additional Web servers and configure them for network load balancing (NLB), possibly with Application Center 2000. The ISA Server then would have to allow port 1433 traffic from all the servers used in the NLB cluster.

XML Web Filters

One of the protocols that ISA Server can filter is HTTP. Often HTTP is allowed through firewalls without inspection, and it recently has become more of a universal protocol for communication among applications. For example, BizTalk Server (see Chapter 12, "BizTalk Server") uses HTTP to send the XML that enables business–to–business (B2B) transactions.

While HTTP may be an easier transport for B2B interactions, hackers target HTTP as a transport to bypass firewall security. This security problem can be alleviated by having ISA Server monitor XML content sent over HTTP before it reaches BizTalk Server computers. ISA Server acts as an intermediary, protecting internal BizTalk Server computers by scrubbing XML traffic before it is allowed through to those computers.

To meet the specific security and performance needs, ISA Server includes a comprehensive Software Developer Kit (SDK). The SDK includes a full Application Programming Interface (API) and many sample filters. Developers can use the SDK to write Web and application filters for ISA Server to monitor content streams. On the basis of the criteria programmed into the filter, various actions can be executed when defined thresholds are crossed.

ISA Server has the application–level filtering capabilities necessary to meet the security demands of the B2B environment. Its comprehensive SDK makes it easy to write custom filters such as the XML filter detailed in this section. As XML data become increasingly widespread, the ability to verify and filter those data is necessary to maintain a secure environment. ISA Server's Web filters offer a sophisticated approach to application–level filtering and allow a high level of customization. The XML filter, which is used in conjunction with the Secure Sockets Layer (SSL) offloading ability of ISA Server, demonstrates that ISA Server can scale to meet an organization's business needs. Custom ISA Server Web filters also do not create a significant load on the computer. They enhance security

without reducing ISA Server performance. The application−level filtering capabilities of ISA Server, combined with the integration capabilities of BizTalk Server, create a secure and functional B2B architecture. The following example describes an XML filter.

The IIS File Drop Version posts the XML data to an IIS computer with an Active Server Page (ASP) page that serializes the XML content into a file for BizTalk Server to retrieve. ISA Server communicates with IIS over HTTP to post the XML, which can be used to traverse firewalls in a DMZ. Certificates are not required for the internal IIS computers. BizTalk Server can use the message queue receive functions or the HTTP receive functions to acquire the XML data. It also can be configured to monitor a folder for new XML files with a file receive function. An XML filter can be written to inspect port 80 traffic. In this scenario you want port 80 traffic to pass through the ISA Server if it is used by BizTalk Server to acquire XML data. You probably do not want all port 80 traffic to pass. The XML Web filter is looking for specific XML attributes in addition to port 80 as the transfer request.

Secure Server Publishing

An improved feature of ISA Server 2000 is the ability to publish internal resources to external clients. With this model both the application Web server and the database server reside behind the firewall. Whether the server is a Web server or a mail server, these resources can be published to the Internet securely and easily. It does not matter if the organization has one ISA Server or several in an array; they all can redirect incoming requests for resources back to the correct server and only that server. Administering server publishing is easy, although some things have to be configured differently. When publishing a server to the Internet, the developer should be prepared to supply the following information:

- **A rule name.**
- **A destination set.** This is a definition of the computers that are affected by this rule. You can include all destinations or only a subset of computers.
- **Client type.** This is a set of users, groups, or IP addresses that you define.
- **Name and IP address of the internal server that you want to publish.**
- **The port you want to connect to.** By default ISA Server tries to connect to port 80 for HTTP, port 443 for SSL, and port 21 for File Transport Protocol (FTP).

The tools for Web publishing and server publishing are easy to use. They are both wizard−driven and need only the specific server name and IP address information to be configured. To start one of the publishing wizards, perform the following steps:

1. Open ISA Management from the Microsoft ISA Server program group.
2. Click to expand Servers and Arrays.
3. Click to expand your server or array.
4. Click to expand Publishing Rules.
5. Right−click the type of publishing rule you would like to create and select New Rule, as shown in Figure 14.6.
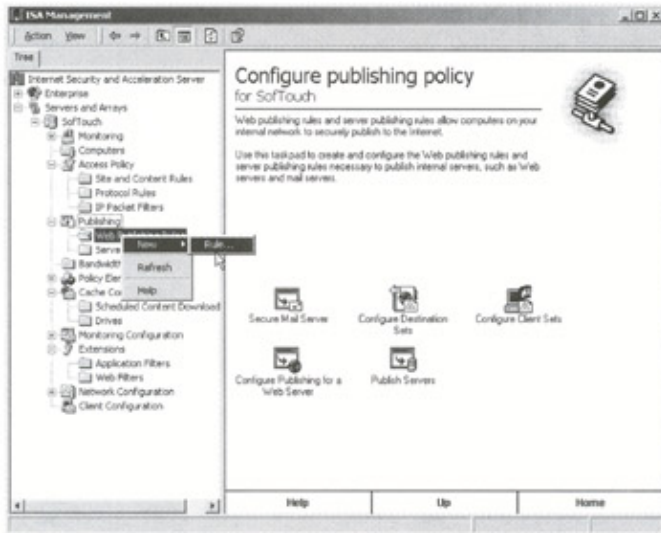
Figure 14.6: Web publishing rules can be used to publish internal Web servers to external (Internet–based) clients. The ISA Server redirects the Internet request to the internal server.

6. Follow the steps laid out by the wizard to publish your resource.

In the Web publishing rules a default rule discards any request for any destination from anyone. If this is not changed, all Web requests that are directed to this server will be discarded. It is vital to change the records in Domain Name Server (DNS) so that users on the Internet can still connect to the Web site.

For example, say your company has a Web site for the general public. The company domain name is registered, and there is a public IP address for your Web server as well. You have just purchased an ISA Server and want to publish the Web server through that ISA Server. You set up a rule on the ISA Server that allows access for any user to just your Web server and alter the internal port numbers to point to ports different from the default ports. You have configured the Web server to listen on the new port numbers. However, when you set everything up, you immediately get calls and email from users who say that your Web site is not up. You check, and nothing is wrong with your Web server.

What is wrong? Did you make sure DNS was pointed to your ISA Server? If it is, the server should be working. Make sure that DNS is pointing to your ISA Server. If you get a new public IP address from your Internet service provider (ISP) or are using a different IP address for your ISA Server, you will need to change your host record in DNS. You could use the same IP address that your Web server was using. It will not need a public IP address now that it is inside your network.

ISA Server does not really redirect incoming requests. Instead, ISA Server acts as a proxy for them just as it does for its internal users. It does not let external users out to the Internet but goes out for them and gets the requested content if it is allowed to. The same rule applies for incoming requests. It does not let external users in but goes and gets the request for content if it is allowed to. The ISA Server isolates the machine from the Internet user. The Web client never has direct access to the internal server.

Note Filtering of SMTP traffic can be set up on an ISA Server so that any SMTP traffic that passes the server can be filtered for content or attachments and even for file extensions. Once these things are filtered, the ISA Server can be told what to do with the content or files it has filtered. The options available are discard, hold, and redirect the filtered information to a different location to be reviewed at a later time.

Enterprise Policies

Enterprise and array policies are used to configure multiple ISA Servers as a single unit. Enterprise policies in ISA Server are configured at the headquarters level of an organization and may apply to any or all arrays. Array policies are configured at the branch level and may inherit an enterprise policy. For any given array, ISA Server allows the user to apply enterprise policies, array policies, or both. Enterprise policies are configured and applied to ISA Server arrays by enterprise administrators. For any given ISA Server array, enterprise policy settings determine whether array policies are included and, if so, what kinds of rules are included in the array policy.

Whenever you install ISA Server after the enterprise has been initialized, you are given the option to install ISA Server as an array member. If you are joining an existing array, the enterprise policies of the existing array are inherited and there is very little configuration at installation time. Enterprise policies include site and content rules and protocol rules. This enterprise policy can be applied to any available array. You can restrict the enterprise policy further by also applying an array policy. You also can create policy elements at the enterprise level. These policy elements are used by enterprise–level rules or array–level rules.

Though by default only users who are members of the Enterprise Admins group have permission to configure enterprise policies, the properties of each enterprise policy can be modified to give other users privileges to change the policy. In other words, an enterprise administrator can give any user or group permission to modify any enterprise policy. The properties of a server's policies can be configured from the ISA Management tool. The properties of the server you want to configure store the settings related to the implementation of the enterprise policies. The following settings can be configured for an enterprise policy, as shown in Figure 14.7:

- **Enterprise policy only.** This setting is configured by selecting the Use This Enterprise Policy radio button and selecting a particular policy. In this case the administrator at the enterprise level dictates that only the selected enterprise policy applies. No new protocol rules or site and content rules can be added at the array level.
- **Combined enterprise and array policy.** This setting is configured by selecting the Allow Array–Level Access Policy Rules That Restrict Enterprise Policy checkbox along with the Use This Enterprise Policy radio button. If this setting is configured, array–level site and content rules, along with protocol rules, are allowed. However, these array–level rules can include only deny–type rules. That is, the protocol rules and site and content rules configured for the array can impose additional limitations only on enterprise–level rules.
- **Array policy only.** This setting is configured by selecting the Use Array Policy Only radio button. In this case no enterprise policy is applied to the array. The array administrator is able to create any rule to allow or deny access. When an array has been configured in this way, the array cannot be configured later to use an enterprise policy.
- **Allow publishing rules.** This setting, which is configured by selecting the Allow Publishing Rules checkbox, controls whether publishing rules may be created on an array. Note that publishing rules themselves cannot be created at the enterprise level.
- **Force packet filtering on the array.** This setting, which is configured by selecting the Force Packet Filtering on the Array checkbox, determines whether packet filtering for an array should be mandatory or made available according to the decision of array–level administrators. Note that IP packet filters can neither be created at the enterprise level nor forbidden by enterprise policy settings.
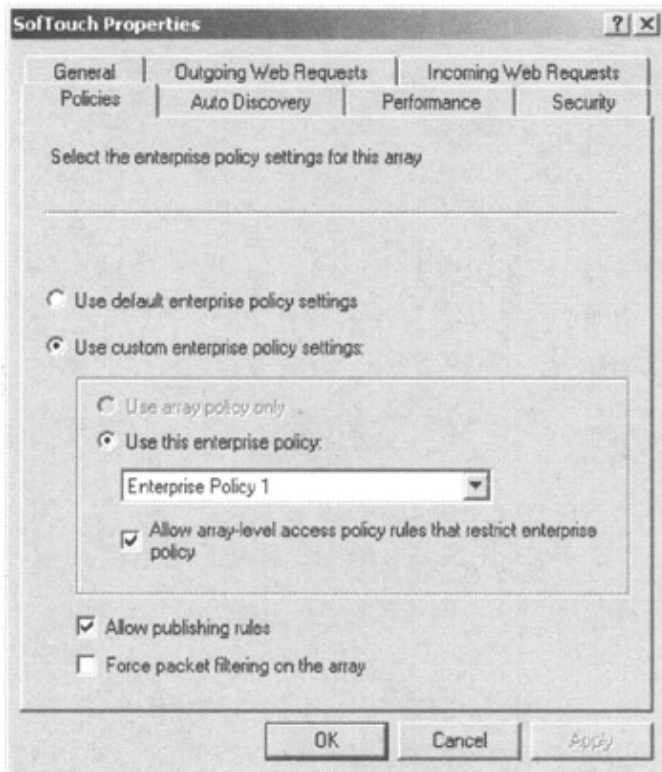
Figure 14.7: Each server contains its settings related to the enterprise policies that are in place.

Integration with Other .NET Enterprise Servers

The previous section outlined some of the aspects of the integration of ISA Server and BizTalk Server. ISA interacts with all the .NET Enterprise Servers as a medium. The ISA Server is a firewall that may have to be traversed for an application to perform its required task. Most of the other products described in this book focus on creating solutions and enterprise applications. This product is focused more on securing the network. Its level of integration is chosen by you as you determine the data flow of your applications and the way in which the applications traverse your firewall structure. As with each of the other .NET Enterprise Servers that has been described in its corresponding chapter, the solutions that Enterprise Servers provide are dependent on interfacing with the Internet as a Web application. ISA Server plays a role in each of them as it serves as the firewall. Server Publishing can play a particularly strong role as you look at the options for placing your Web servers behind the firewall.

For example, your organization could have an employee time−tracking application that feeds employee hours into your human resources and payroll packages. The time−tracking package provides several options for employees to enter their times. You would like to extend this to include a Web interface so that users can log on over an Internet connection and enter their times. This is of particular importance at some of the remote sites where employees are currently punching time cards. At the current time the data are stored in an SQL Server database that is accessed from a variety of front−end applications. You are transferring data using Data Transformation Services (DTS) to transfer the time data out of this database and into your human resources and payroll packages.

You create an application in Visual Basic .NET that uses ASP.NET. The ASPs are placed on a Web server that resides outside your firewall. The ASPs then connect to your SQL Server database, which sits behind the firewall. You have to allow the connectivity to your database. You can configure the port to be open on the firewall to allow the connectivity. The default port for SQL Server is 1433, but that can be changed, and so you should verify the port that is used for the

connection. When you open this port on the firewall, you want to open it as a tunnel so that the only computer that can make the connection to the SQL Server through the firewall is the computer running IIS. This is the Web server that is storing your ASPs.

You have a couple of alternatives to make this application more scalable and increase its availability. Employee time tends to be a critical function, and employees are not happy when your application is not available. You could deploy a second Web server outside the firewall; that server also has to have the ASPs deployed on it. You can manage the synchronization of these files with Application Center 2000. You then can use the NLB services of Application Center to load balance the connections from the Internet. This also helps guarantee an available connection. If one server is down, you still have access through the other server. Your ISA Server then needs to be configured to allow connections to port 1433 for both Web servers outside the firewall.

Note You could help increase the security of this time−tracking package by placing your Web server behind the firewall and using Server Publishing to connect to the server. If you did this, the Web server would not need a valid IP address.

As in this example, ISA Server plays a key role in securing your network. You should design all your Web applications around your firewall strategy. You also should try to implement the features of ISA Server that best facilitate a secure and fast connection to your data.

Review Questions

1. What are the differences between the Enterprise Edition and the Standard Edition of ISA Server?
2. Why would you want to use ISA Server in a Cache−only mode?
3. What is Secure Server Publishing?
4. How does ISA Server interact with the other .NET Enterprise Servers?
5. What is an XML Web filter?
6. Why would you use an enterprise policy, and what is its relationship to an array policy?

Part III: Enterprise Solutions with .NET Servers

Chapter List

Chapter 15: Web Services with .NET

Highlights

With the evolution of XML, Web services have piqued the interest of many organizations. Terms such as SOAP, Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI) have become industry buzzwords.

Web services allow a developer to build loosely coupled applications with a focus on integration. A loosely coupled application is one that consists of several pieces and parts, each of which is not dependent on the others. Developers then assemble the pieces and parts that make up the application. The developer can take any of the pieces and upgrade, redeploy, and reconfigure them without affecting the entire application. Web services can be advantageous pieces of this type of application.

To achieve maximum interoperability, Web services were established on commonly used industry–standard protocols. XML is the foundation for many of those standards. The Web standards that are implemented by XML provide a uniform, platform–independent way to represent data. SOAP builds on to XML by providing a standard by which data can be sent to and from applications.

The transformation typically occurs over HTTP. WSDL describes the signature, or interface, and the other essential parameters for a Web service. UDDI provides a directory in which a user can search for the Web services he or she would like to use or in which to register an existing Web service.

This chapter outlines the role of each one of these buzzwords in Web services. The chapter begins with an overview of Web services to help a user determine when he or she needs Web services and how they can be used with his or her applications. The chapter then discusses the standards on which Web services are built. This section describes to the role of XML and Web services. Then the reader is introduced to SOAP and its role in Web services. The chapter then identifies the purpose of WSDL and UDDI.

After gaining an understanding of the basics of Web services, it is necessary to know how to implement them. The chapter then explains how Web services interact with .NET Enterprise Servers. The chapter includes examples of how Web services can be used within the applications a user has to develop.

Overview of Web Services

A Web service is a set of application logic that is created programmatically and can be called from any application. Web services combine the developmental advantages of desktop development tools with the deployment advantages of the Web. Like .NET components, Web services create logical functionality that can be reused by any application. The application does not depend on the method with which the service was created or implemented.

Web services are accessible via Web protocols such as HTTP and XML. This is among the core differences between Web services and .NET components. .NET components are accessed via object model protocols such as Distributed Component Object Model (DCOM).

Web services are very well defined in terms of standards and structure. Five things concern the consumers of a Web service:

- The standard way in which the data are presented
- The service description language that the service uses
- The message format standard
- How the Web service can be found on the Web site
- How to discover service providers for the service

XML is the most widely used standard for presenting the data. XML is a solid choice because it is supported across many platforms and because of its ease of implementation. The core of this chapter centers on using XML for the creation and representation of data through Web services.

SOAP defines a protocol for the messaging standard. SOAP defines standards for the following items related to a Web service:

- Rules for using XML to represent data
- Rules for the messaging format
- Rules for binding to the HTTP protocol
- Conventions for using remote procedure calls

Microsoft and other authorities anticipate that SOAP will become the standard for communicating with all Web services.

A Web service needs to have documentation on what it will accept and how developers can interact with it. WSDL defines the methods, parameters, and data types used by the Web service. While WSDL was not the first descriptor language, it is the most common.

Developers need to find the Web service. If the Web site is known, a developer can use the Discovery Protocol (Disco) to find a Web service. In many cases the developer will not know the Uniform Resource Locator (URL). UDDI provides a mechanism to publish Web services so that they can be found easily over the Internet.

Now that the reader has been introduced to Web services, it is important to lay a foundation for their use. The following sections address the reasons for using Web services, Web services and .NET, the Web service architecture, Web service development environments, and Microsoft's Web service solution.

Web Service Usage

A developer may wonder what he or she can do with Web services. The original Web services were limited to informational sources that one could add to an application. For example, you may want to post stock quotes or news updates on your Web site. The XML Web service can be used to retrieve that information.

This function has been extended by many organizations to include more dynamic business processes. For instance, if you track your company's financials in an Excel spreadsheet, you can expose the data in a summarized fashion. Excel can continue to update the information and perform the analysis and functions you require. You can retrieve the results of the processing and display them in a Web application report. This can be an effective way to take advantage of some of the more complex functions in Excel, such as amortization, loan payments, and depreciation schedules.

Exposing existing applications as XML Web services allows users to build new, more powerful applications that use XML Web services as building blocks. For example, a user might develop a purchasing application to obtain price information automatically from a variety of vendors, allowing the user to select a vendor, submit the order, and track the shipment until it is received. The vendor application, in addition to exposing its services on the Web, might use XML Web services to check a customer's credit, charge the customer's account, and set up the shipment with a shipping company.

In the future some of the most interesting XML Web services will support applications that use the Web to do things that cannot be done today. Web services can be helpful for the following reasons:

- Virtually all development platforms can interact with Web services.
- Web services are useful for centralizing business logic that will be reused by multiple applications.
- Web services can define business processes and serve as a building block for Web applications.
- Web services are built on open industry–standard protocols that can be used to represent data across any medium.
- Web services can facilitate application interoperability.

Web Services and .NET

The purpose of Web services is to facilitate a loosely coupled architecture for application development. It is anticipated that applications will be created that are tied together with multiple Web services. The Web services can define the logic and data access methods for the application.

In a normal Windows Distributed InterNet Architecture (DNA), the application logic generally is defined in the business layer and the data are stored in the data layer. In many cases they are stored on separate servers. Both layers can be defined by the organization's Web services. In a sense you are taking your Windows DNA architecture and deploying it over the Internet. Because the models are so similar, the development of applications that use Web services is not very different from the use of the Windows DNA. The traditional Windows DNA model consists of three logical layers, as shown in Figure 15.1. For more information about each of these layers, refer to Chapter 16, "Building Enterprise Web Farms."

Three-Layer Application Architecture

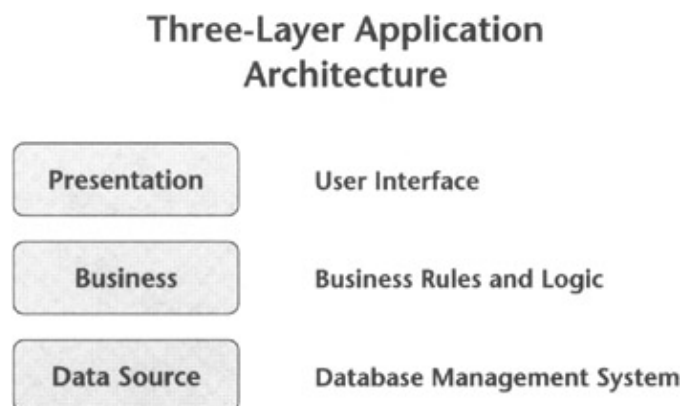| Presentation	User Interface
Business	Business Rules and Logic
Data Source	Database Management System

Figure 15.1: The traditional Windows DNA model consists of a logical three–layer architecture.

Web Services Application Architecture

As you develop Web services, you will be concerned about the internal makeup of the service. From an application development perspective, the internal workings are not critical. The application developer is more concerned with the external view of the Web service. That view consists of the

availability, usability, and functionality of the Web service. The application developer only needs to be able to use the Web service as a dependable element of the application.

The architecture for a Web service is divided into five layers, as shown in Figure 15.2. The layers are described in more detail in the following list:

- The *listener* is the layer of the application that is closest to the user. It is responsible for receiving an incoming request and routing it to the business façade. If the Web service has to send a response to the user, the listener is also responsible for packaging the request in the appropriate format for the user.
- The *business façade* layer provides an interface to map the services that are defined in the business logic layer. In the traditional Windows DNA, the business layer is responsible for the business logic. In the Web service architecture, this responsibility typically is broken into two separate layers. The business façade provides the translation between the listener and business logic.
- The *business logic* layer includes the core services that provide the business logic of the application.
- The *data access* layer provides a logical view or representation of the data to the business logic layer.
- The *data* layer is the farthest from the client. This is where the data are stored. In most cases this layer is the relational database management system.

Web Service Architecture
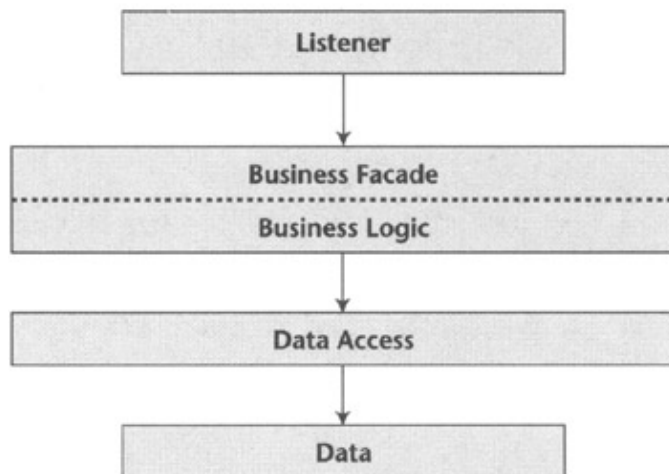
Figure 15.2: Web services have a logical five–layer architecture.

This architecture is similar to the *n*–tier application architecture defined by Windows DNA. As shown in Figure 15.3, the Web service listener is equivalent to the presentation layer of a Windows DNA application. It is common for an application to add a Web service to its existing architecture. This can be done by providing a listener for an existing business layer, which is equivalent to the business façade and business logic of the Web service architecture.
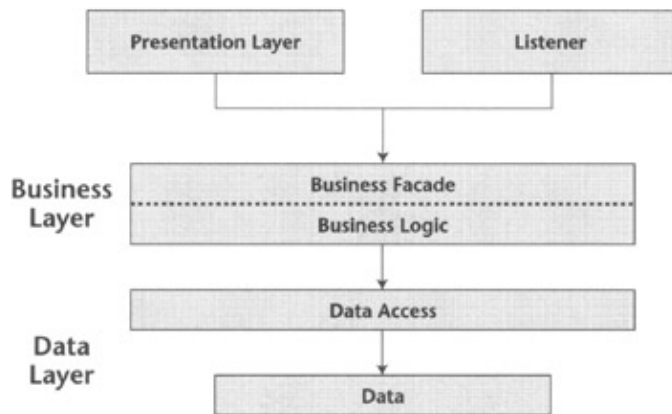
Three-Layer Architecture

Figure 15.3: A listener can be added to an existing application and plays a role similar to that of the presentation layer.

In the traditional Windows DNA one is used to thinking about implementing the data layer by using databases and implementing the business layer by using COM+ components. But what if the data access layer gets its data from a Web service that accesses other underlying data? What if the business facade calls a Web service to perform a portion of its work? In this case the application architecture starts to look a lot like the Web services architecture rather than the traditional Windows DNA. The Windows DNA has been extended to include the advantages provided by the Web services architecture.

Development Environments

In light of the similarities between the Windows DNA and the Web services architecture, it should not come as a surprise that developing Web services is not very different from developing Web applications. A developer also can develop Web services that are used by traditional Windows DNA applications.

There are some differences, however. In creating Web services it is necessary to account for the following items, which are not part of the traditional Windows DNA model:

- You need to understand SOAP. You have to understand SOAP messages and generate responses by using SOAP.
- You need to create a WSDL contract. A WSDL contract defines the methods, parameters, and data types that are used by the Web service.
- You should create a Disco file for the site. This allows developers to find the Web services on your site. It is helpful for developers who already know the location of your Web site.
- You also should consider advertising your service through UDDI. UDDI advertises your service via the Web. It is necessary if you want developers who do not know the URL of your Web site to find your Web service.

In many cases Web services become part of the applications created by other developers. Because applications are dependent on your service, you want that service to have the following features:

- **The Web service should be dependable.** The service should be available at all times. Applications depend on the availability of your service. For instance, if you create a Web service that validates credit card authorizations, it would be a severe limitation to your clients if the service was not readily available for an extended period of time.
- **The Web service should be easy to deploy.** In many cases your Web service may be

213

deployed by an Internet service provider (ISP) because the provider may have a higher bandwidth and a better internal infrastructure (such as servers, backup, disaster recovery, and redundancy). The ISP may not deploy the service if it is difficult to install.

- **The Web service should be easy to maintain.** You should have a strategy to upgrade, redeploy, and manipulate the Web service. For critical services, you should consider using network load balancing (NLB) to help maintain the Web service. With NLB the Web service will be installed on multiple servers, with the requests alternating among the servers. You can take a server offline and upgrade, redeploy, or reconfigure the Web service on that server. While you are performing the maintenance, activity can occur against servers that are still online. This helps guarantee that your service is always available.
- **The Web service should be responsive.** If you expect other applications to use your Web service, you should take the appropriate measures to ensure that the service is fast. You should provide the appropriate level of bandwidth. You may want to consider hosting the service at an ISP to provide the bandwidth necessary for your Web service. You also could consider using NLB to help with performance. In many cases, performance problems are caused by a lack of bandwidth. NLB will not help with lack of bandwidth issues. It will help, however, when a single server is the source of the bottleneck. If you are receiving too many requests to handle the incoming traffic, additional servers in an NLB scenario help distribute the load and increase performance.

If you plan to market a Web service to other organizations, you want to make that Web service easy for those organizations to use. You do not want to create an undue burden on the developers in the other organizations. Web services that are marketed and sold should be designed with other developers in mind.

In many cases this includes creating middleware that resides at your location. Middleware is an application that provides the tools needed for developers to use, troubleshoot, test, and tune their applications. In general, the middleware should be an application that helps other developers interact with your Web service during their design time. Keep in mind that you want your Web service to be easy to use. If it is difficult to use, most organizations will not want to use it. Your middleware should be responsible for the following items:

- Listening for incoming HTTP requests.
- Performing security authentication and authorization checks.
- Dispatching authorized requests to the correct service.
- Ensuring that services are isolated from one another and the hosting environment, meaning that each service has its own memory, services cannot block other services from executing, and service faults cannot cause other services or the hosting environment to default.
- Automatically recovering from service, hosting environment, and system failures.
- Providing administrative facilities for deploying, monitoring, and controlling services.
- Managing resources such as processes, threads, and shared state on behalf of each service.

Including appropriate middleware Application Programming Interfaces (APIs) for constructing and parsing messages also enhances your productivity. At a minimum, APIs must support reading and writing XML streams. Specific APIs for standards such as SOAP, WSDL, Disco, and UDDI improve productivity as well as overall reliability by eliminating the need to write parsing and formatting logic. APIs also reduce the need to learn every detail of the specifications.

If you are implementing Web services using .NET components, additional productivity gains can be achieved if the system provides services to activate objects on demand. Additionally, components could map messages to object method calls, which essentially implement the Web service listener

for you.

Microsoft Web Service Solution

The .NET Server platform provides the necessary foundation for the development of Web applications and Web services. .NET Internet Information Server (IIS) and Component Services (COM+) provide the infrastructure for Web application and service development. The .NET Server platform also provides APIs to assist in all layers of the application architecture. Even though you are implementing the application through Web services, you still can use the common interfaces to perform the application development requirements. The following list outlines the supported technologies for various application needs:

- .NET components for implementing the business façade, business logic, and data access layers
- ActiveX Data Objects (ADO), Object Linking and Embedding Database (OLE DB), and Open Database Connectivity (ODBC) for implementing data access to a variety of data stores
- MSXML (the parser for XML) to help construct and consume XML messages in the Web service listener
- Active Server Pages (ASP.NET pages) to implement the Web service listener

Web Service Standards

XML Web services open the door to distributed computing over the Internet. It is possible to create applications that invoke Web services in a manner similar to the way in which the Window DNA model has used Component Object Model (COM) objects and Windows services. The difference is that Web services are easily accessible over the Internet. It is possible to access the Web service from any application through the connectivity of the Web. The development of several industry–standard open interfaces and protocols has created an environment in which XML Web services are available for application integration. While all Web services may provide different services, each one implements some common rules and standards. These standards were presented in the earlier section of this chapter titled _Overview of Web Services_.

One of the primary advantages of the XML Web services architecture is that it allows programs written in different languages on different platforms to communicate with each other in a standards–based way. Those who have been developing for a while undoubtedly have heard this before. Why are XML Web services different? There are two primary reasons:

- XML Web services are built on Web standard protocols: XML, HTTP, and Transition Control Protocol/Internet Protocol (TCP/IP).
- SOAP is less complex than earlier messaging protocols.

This chapter has defined an XML Web service as a software service exposed on the Web through SOAP, described with a WSDL file, and registered in UDDI. SOAP, WSDL, and UDDI are described in more detail in the following sections.

SOAP

Soap is the communications protocol for XML Web services. When SOAP is described as a communications protocol, most people think of DCOM or Common Object Request Broker Architecture (CORBA), and this begs the question in relation to object activation and naming

services. Your SOAP implementation may include these items, but the SOAP standard does not specify them for you. SOAP is a standard that defines the XML format for messages that are sent and received from the Web service. If you have a well-formed XML fragment enclosed in a couple of SOAP elements, you have a SOAP message. It is that simple.

Other parts of the SOAP specification describe how to represent program data as XML and how to use SOAP for remote procedure calls (RPCs). These optional parts of the specification implement RPC-style applications in which a SOAP message containing a callable function and the parameters to pass to the function is sent from the client. The server returns a message with the results of the executed function. Most current implementations of SOAP support RPC applications because programmers who are used to COM or CORBA applications understand the RPC style. SOAP also supports document-style applications in which the SOAP message is only a wrapper around an XML document. Document-style SOAP applications are flexible, and many new XML Web services take advantage of this flexibility to build services that would be difficult to implement by using RPC.

The last optional part of the SOAP specification defines what an HTTP message that contains a SOAP message looks like. This HTTP binding is important because HTTP is supported by almost all current Operating Systems (and many not so current OSes). The HTTP binding is optional, but almost all SOAP implementations support it because it is the only standardized protocol for SOAP. For this reason, there is a common misconception that SOAP requires HTTP. Some implementations support message queuing (MSMQ), MQ Series, Simple Mail Transport Protocol (SMTP), or TCP/IP transports, but almost all current XML Web services use HTTP because it is ubiquitous. Since HTTP is a core protocol of the Web, most organizations have a network infrastructure that supports HTTP and people who understand how to manage it. The security, monitoring, and load-balancing infrastructures for HTTP are readily available.

A major source of confusion when one is getting started with SOAP is the difference between the SOAP specification and the many implementations of the SOAP specification. Most people who use SOAP do not write SOAP messages directly; instead, they use a SOAP Toolkit to create and parse SOAP messages. These toolkits generally translate function calls from some kind of language to a SOAP message. For example, the Microsoft SOAP Toolkit 2.0 translates COM function calls to SOAP and the Apache Toolkit translates JAVA function calls to SOAP. The types of function calls and the data types of the parameters supported vary with each SOAP implementation, and so a function that works with one toolkit may not work with another. This is not a limitation of SOAP but a limitation of a particular implementation.

By far the most compelling feature of SOAP is that it has been implemented on many different hardware and software platforms. This means that SOAP can link disparate systems within and outside an organization. Many attempts were made in the past to come up with a common communications protocol that could be used for systems integration, but none had the widespread adoption that SOAP has had. This is the case because SOAP is much smaller and simpler to implement than are many of the earlier protocols. The following sections identify the additional concerns for SOAP security and the tools available for implementing SOAP.

SOAP Security Concerns

One of the first questions newcomers to SOAP ask is how SOAP deals with security. Early in its development, since SOAP was seen as an HTTP-based protocol, it was assumed that HTTP security would be adequate for SOAP. After all, thousands of Web applications running today use HTTP security, and so surely it is adequate for SOAP. For this reason the current SOAP standard assumes that security is a transport issue and is silent on security issues.

When SOAP expanded to become a more general–purpose protocol running on top of a number of transports, security became a bigger issue. For example, HTTP provides several ways to authenticate which user is making a SOAP call, but how is that identity propagated when the message is routed from HTTP to an SMTP transport? SOAP was designed as a building–block protocol, and so specifications already in the works build on SOAP to provide additional security features for Web services. The reader can view the complete WS–Security specification to get a definition of the encryption system used by SOAP at http://msdn.microsoft.com/library/en–us/dnglobalspec/html/ws–security.asp.

SOAP Tools

Developers who are creating SOAP–based Web services have three options:

- Create their own by using MSXML or ASP.NET. The downside, of course, is that the developer has to implement and test everything himself or herself and figure out how to comply with the relevant specifications.
- Use the SOAP Toolkit for Visual Studio .NET to build a Web service listener that connects to a business façade implemented by using COM. (The SOAP Toolkit can be used if the business façade is not implemented as a COM component, but the wizard that generates the listener cannot be leveraged as easily.) Note that the SOAP Toolkit is a sample provided by Microsoft Solution Developer Network (MSDN). The toolkit understands SOAP over HTTP and SSL but does not help the developer create Disco documents or UDDI registrations. It supports an older contract language called Services Description Language (SDL) rather than WSDL.
- Use the Microsoft Soap Toolkit Version 3.0 to build a Web service listener that connects to a business façade implemented by using COM. Version 2 supports SOAP over HTTP and can create a WSDL file describing the service. It still is necessary to create Disco documents and UDDI registrations manually.

Both versions of the SOAP Toolkit provide tools that help developers consume Web services as if they were COM components. The SOAP Toolkit provides a COM component called the Remote Object Proxy Engine (ROPE) that can be used by client applications. ROPE uses an SDL file to dynamically create automation methods that can be called on a proxy object. If you want to use a Web service that does not supply an SDL file, you have to create one. Version 2 of the SOAP Toolkit provides similar functionality that is based on WSDL files.

For additional information on implementing SOAP, visit the SOAP homepage at the MSDN Web site at http://msdn.microsoft.com/library/default.asp?url=/nhp/default.asp?contentid=28000523.

WSDL

For the purposes of this book, a WSDL (pronounced whiz–dull) file is an XML document that describes a set of SOAP messages and shows how the messages are exchanged. Since WSDL is XML, it is readable and editable, but in most cases it is generated by software.

To see the value of WSDL, imagine that you want to start calling a SOAP method provided by one of your business partners. You could ask the partner for sample SOAP messages and write your application to produce and consume messages that look like the samples, but this can be an error–prone process. For example, you might see a customer number of 2002 and assume that it is an integer when in fact it is a character data type. WSDL specifies what a request message must contain and what the response message will look like in unambiguous notation. This helps avoid data type and missing parameter errors.

The notation a WSDL file uses to describe message formats is based on the XML schema standard; that means it is both programming–language–neutral and standards–based, making it suitable for describing XML Web services interfaces that are accessible from a wide variety of platforms and programming languages. In addition to describing message contents, WSDL defines where the service is available and which protocol is used to communicate with the service. This means that the WSDL file defines everything required to write a program that will work with an XML Web service. Several tools can read a WSDL file and generate the code required to communicate with an XML Web service. Some of the most capable of these tools are in Microsoft Visual Studio .NET.

Many current SOAP Toolkits include tools that are used to generate WSDL files from existing program interfaces, but at the time of this writing few tools can write WSDL directly and tool support for WSDL is not as complete as it should be. It should not be long before tools to author WSDL files and then generate proxies and stubs much as COM Interface Definition Language (IDL) tools do are part of most SOAP implementations. At that point WSDL will become the preferred way to author SOAP interfaces for XML Web services.

There is a more detailed description of the WSDL specification at www.w3c.org/TR/wsdl.

UDDI

Universal Discovery, Description, and Integration is the Yellow Pages of Web services. As with the traditional telephone Yellow Pages, a user can search for a company that offers the services he or she needs, read about the services offered, and contact someone for more information.

One can, of course, offer a Web service without registering it in UDDI, just as one can open a business in one's basement and rely on word–of–mouth advertising. However, if you want to reach a significant market, you need UDDI so that your customers can find you.

A UDDI directory entry is an XML file that describes a business and the services it offers. There are three parts to an entry in the UDDI directory. The "White Pages" describe the company offering the service: name, address, contacts, and so forth. The "Yellow Pages" include industrial categories based on standard taxonomies such as the North American Industry Classification System and the Standard Industrial Classification. The "Green Pages" describe the interface to the service in enough detail so that one can write an application to use the Web service. Services are defined through a UDDI document called a type model, or tModel. In many cases the tModel contains a WSDL file that describes a SOAP interface to an XML Web service, but the tModel is flexible enough to describe almost any kind of service.

The UDDI directory also includes several ways to search for the services needed to build applications. For example, you can search for the providers of a service in a specified geographic location or for businesses of a specified type. The UDDI directory then supplies information, contacts, links, and technical data that allow you to evaluate which services meet your requirements.

UDDI allows you to find businesses that provide Web services. What if you already know who you want to do business with but do not know what services are offered? The WS–Inspection specification allows you to browse through a collection of XML Web services offered on a specific server to find which ones meet your needs. The WS–Inspection specification is available at http://msdn.microsoft.com/library/en–us/dnglobspec/html/ws–inspection.asp.

More information about UDDI is available at www.uddi.org/about.html.

Web Services and .NET Business Solutions

.NET Enterprise Servers provide a platform on which to deploy and interact with Web services. At the time of this writing some .NET Enterprise Servers, such as Exchange Server and BizTalk Server, do not support Web services directly. They do, however, support XML and SOAP and can be used to create distributed Web applications that are similar to Web services. A couple of the servers—SQL Server and Application Center 2000—are particularly beneficial to Web service development. The following list defines these two servers from the perspective of their role in Web services:

- Application Center 2000 deploys and manages Web applications and Web services. Of particular interest are the NLB and synchronization services. Application Center 2000 can be used to increase the stability and availability of a Web service.
- SQL Server 2000 includes extensive support for XML. Relational data can be queried and modified as XML, eliminating the need to handcraft formatting logic in one's applications. Through SQLXML 3.0 one can expose a stored procedure as a Web service.

The following sections describe the role of these two servers with Web services in more detail. They include some examples of how these servers can benefit applications. After introducing the role of Application Center 2000 and SQL Server with Web services, the section concludes with a short description of how Web services can be used to create enterprise application integration (EAI) solutions, business–to–business (B2B) solutions, and business–to–consumer (B2C) solutions.

Application Center 2000 and Web Services

Application Center 2000 is a server tool that enhances the performance and functionality of Web sites that are deployed on IIS. As was mentioned earlier in this chapter in the section titled _Development Environments_, it is possible to increase the dependability of a Web service by deploying the service across multiple servers that implement NLB. With the standard NLB service of .NET Server, it is necessary to maintain the files that are deployed on all the servers in a cluster. You want all the servers in the server cluster to have identical files for the Web service. You cannot have Web users see different content or functionality on a random basis. Application Center 2000 provides a synchronization service that can keep all the servers in a cluster up to date with the most recent version of the Web service.

Application Center 2000 also can be used to take servers offline and place them back online. This feature facilitates taking a server down to perform maintenance against the server without affecting the functionality or availability of the Web service.

SQL Server and Web Services

You can extend the functionality of your stored procedures in SQL Server by exposing them as Web services. You can use Transact–SQL to create your stored procedures as you do with your other stored procedures and then, through the integration with XML, expose a stored procedure as a Web service. Then any application can connect to your service and use the functionality of the stored procedure. This process can be beneficial in making stored procedures accessible from any platform. You can create multiple Web applications that use the same Web services to accomplish a task.

For example, say you are a sales organization. You have a fairly complex commission algorithm that computes commissions on a specific product. You may have a couple of Web applications that need to use this. You could have one application for internal sales staff and another for partner

resellers. You can write the logic in a stored procedure and then expose the procedure as a Web service. You then can access the Web service from your Web applications by using SOAP.

This section describes the process of creating a Web service from a stored procedure through the use of SQLXML 3.0 and then discusses the process of using the Web service from your application.

Creating a Web Service

To facilitate the process of creating a Web service from an SQL Server stored procedure, you should have the SQL Server XML Toolkit installed on your machine. For more information on the installation and purpose of the toolkit, refer to Chapter 8, "SQL Server 2000." Through the installation of the toolkit, you installed SQLXML 3.0. The primary change from Version 2.0 to Version 3.0 is the support for Web services.

Creating a Web service from a stored procedure involves several steps:

1. Create the stored procedure.
2. Configure the virtual directory.
3. Configure the virtual name.
4. Create the Web service.

Each of these steps is described in more detail in the following sections.

Creating the Stored Procedure

You should create your stored procedure in the same way you create any other procedure. You can create it from a query tool that has the ability to execute transact–SQL against your SQL Server.

The most common interface for stored procedure creation is Query Analyzer, the query tool that ships with SQL Server 2000. In these examples the code is presented in transact–SQL format, which should work from any query tool that is being used for SQL Server 2000. The following code creates a stored procedure that selects data from the pubs database. The pubs database is a built–in database for SQL Server that is provided for testing. The example is simplistic but demonstrates the process:

```
CREATE PROCEDURE sEmployeeList
AS
SELECT * from Employees
GO
```

Note Transact–SQL and the creation of stored procedures are not core topics of this book. For more information on the details of these items, refer to *SQL Server Books Online*, which ships with SQL Server 2000.

Configuring the Virtual Directory

For your SQL Server data and objects to be accessible from the Web via XML, you need to configure a virtual directory to facilitate the connection. You should perform this configuration from the SQL Server tools rather than from your standard IIS administrative tools. Before configuring the virtual directory, bear in mind the following information:

- The name you would like for your virtual directory.
- The SQL login account that has permission to access the database where the stored procedures are located.

- The database where the stored procedures are located.
- Whether you need users to POST data back to your database. This is important for Web services but is not needed for other XML and SQL Server requests.

After you are familiar with the required elements, you are ready to configure the virtual directory. To do this, perform the following steps.

1. Open the Configure SQL XML Support in the IIS tool from the SQL Server program group.
2. From the Microsoft Virtual Directory Manager, as shown in Figure 15.4, click to expand the Web server.
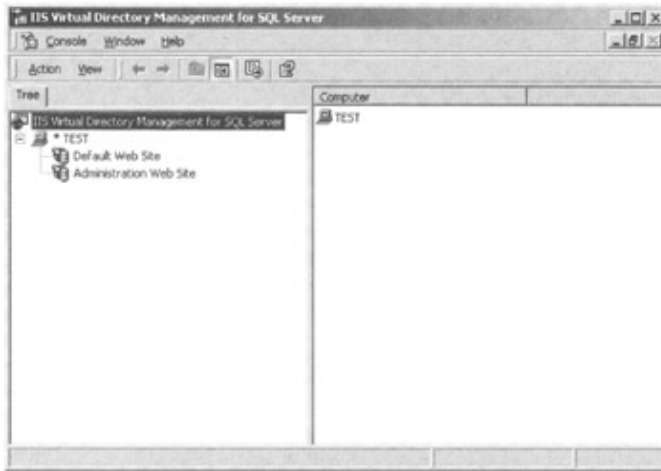
Figure 15.4: The Microsoft Virtual Directory Manager configures a virtual directory for the integration of SQL Server and XML.

3. Right–click the default Web site and select New Virtual Directory.
4. From the General tab, give the virtual directory a name.
5. Also from the General tab, select the local system directory.
6. On the Security tab, enter the login name of a user who has access to the database where the stored procedure resides.
7. On the Data Source tab, select the database that should be used to search for the appropriate data and objects.
8. On the Settings tab, select the Allow POST option, as shown in Figure 15.5, and then click Apply.
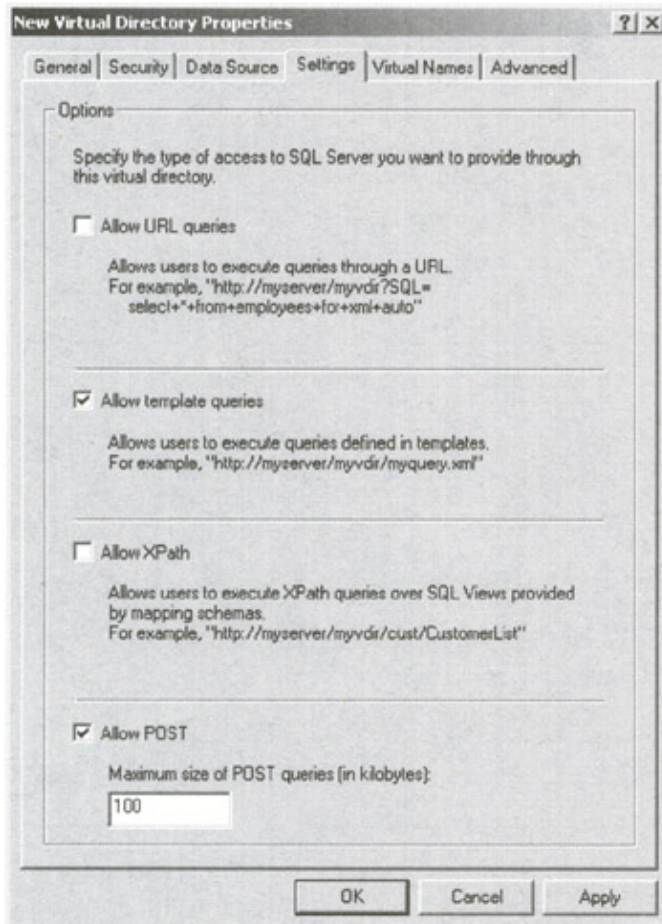
Figure 15.5: Click Allow POST so that the virtual directory will accept HTTP POST requests.
9. Click OK to close the dialogue box.

For more information on the other options in the Settings tab, refer to Chapter 8, "SQL Server 2000."

Configuring a Virtual Name

After configuring the virtual directory, you have to configure the virtual name. This is different from the name of the virtual directory. The virtual name is used in the SOAP string to request a connection point. When you create the virtual name, you also must supply a path. This can be the same as the virtual directory path, or it can be another directory on your system. When you save the changes to your virtual name, the path contains the WSDL output file, which contains the data types, location, methods, and parameters for the Web service.

When you save the virtual name, SQLXML 3.0 creates two files in the directory you chose for the virtual name. One file, which has a .ssc file extension, contains the XML description for the stored procedures. The other file, which has a .wsdl file extension, contains the descriptive WSDL information for your Web service. Your stored procedures are treated as methods for your Web service.

Creating a Web Service

Creating a Web service is not a difficult process after the virtual directory and virtual name have been configured. You need to define the stored procedures that you want to expose as methods for your Web service. You should choose stored procedures that return standard result sets.

You do not have to use procedures that use the FOR XML clause. SQLXML 3.0 handles the conversion to XML process, and so your procedures do not have to account for the process. This provides you with the most flexibility in choosing stored procedures, and your performance of the procedure will be quicker.

To expose your stored procedure as a Web service, perform the following steps. These steps include the example stored procedure (sEmployeeList) that was created earlier in the pubs database. You will have to replace sEmployeeList with the name of your procedure:

1. Open the Configure XML Support for SQL Server tool from the SQL Server program group.
2. From the Microsoft Virtual Directory Manager, click to expand the Web server.
3. Click to expand the default Web site.
4. Right–click the virtual directory and select Properties, as shown in Figure 15.6.
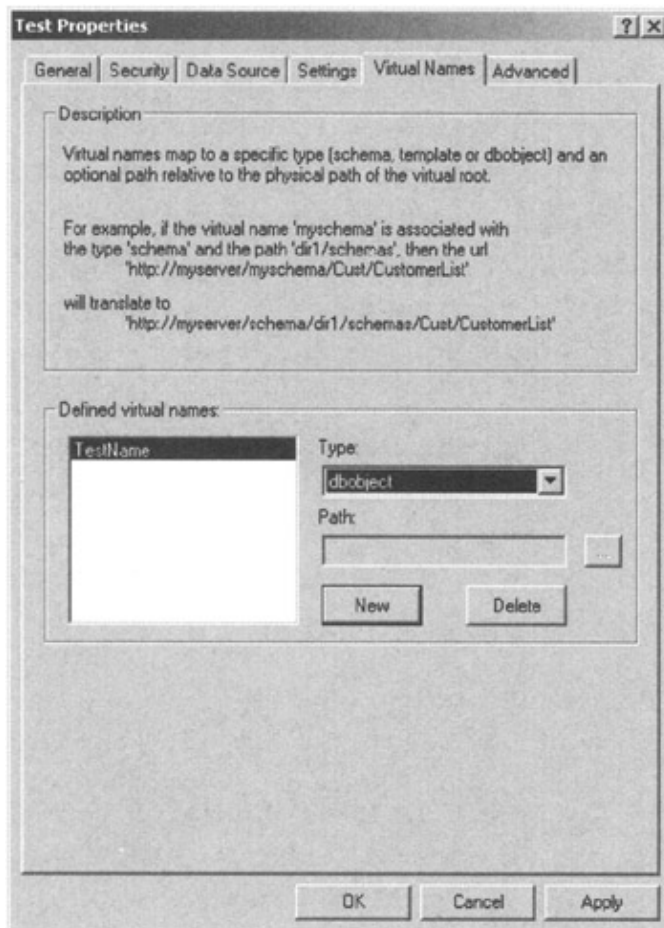
Figure 15.6: Through the properties of the virtual directory it is possible to configure the virtual names for a Web server.

5. Click the Virtual Names tab.
6. Click the virtual name and select Configure.
7. From the SOAP virtual name configuration dialogue box, select SP as the type of method mapping.
8. From the SP/Template field, click the ellipses to bring up a list of available stored procedures.
9. Select the sEmployeeList stored procedure and then click OK.
10. Accept the rest of the defaults by choosing OK to close the dialogue box.

After you have completed these steps, the stored procedure is exposed as a method for your Web service.

Creating Web Applications That Use Your Web Service

Now that you have created a Web service that exposes an SQL Server stored procedure, you need to understand the steps used to access the Web service from the applications you are creating.

Say your company uses a purchase order application that stores vendor−specific purchase orders in an SQL Server database. The database also tracks the status of the purchase order. Your vendors do not have access to the status data. When they want to know the status of a purchase order, they have to call your organization. You would like to cut down on those calls and save time for your employees. You want to create a method for your vendors to access the data directly and integrate the results into their own systems. You need to provide your vendor with an interface to the data (Web service). You also need to agree with your vendor on the format in which you will return the data. The format includes considerations for data types, parameters, methods, and so forth.

This example is an ideal candidate for a Web service. You could use an XML Web service that is SOAP−based to provide the vendors with the data they require. You could create a stored procedure that retrieves the necessary data and expose the stored procedure as a Web service.

When you created the virtual name for your Web service, as was shown in the section titled *Creating a Web Service*, the IIS Virtual Directory Manager created a WSDL file in the directory path you identified. Your client applications use the WSDL file to get the descriptive information pertaining to your service. The WSDL file is critical for the Web applications that invoke your Web service.

WSDL describes each method as a set of messages that the client and the server need to exchange. The WSDL file is an XML document that contains the following information for the client:

- Parameter names and parameter data types
- URL to invoke the method
- Return values

The WSDL file describes the messages by using elements. Elements are presented in the wsdl:message format. Each message contains a message part that represents a complex XML. The complex XML is specified by using the wsdl:part subelement. The part's subelement refers to a complex type that is defined within the wsdl:types elements in the same WSDL file. SQLXML 3.0 defines the complex types by using the standard XML Schema Definition (XSD) schemas. Each WSDL file contains several wsdl:types that represent the common types that are used for many Web services.

Your Web service defines its additional schemas by using import statements. For example, if you create a stored procedure named getPOInfo that a vendor can use to retrieve purchase order information, you most likely will create a couple of complex types. In the file you can see the getPOInfo messages. You most likely will see getPOInfo and getPOInfoResponse complex types. These type definitions specify the XML content of the messages that pass between the Web service and the Web application user.

The client can use the WSDL file to interpret the SOAP format necessary for your Web service. The client can pass the properly formatted message to SQLXML 3.0, which interprets the message and

executes the underlying stored procedure. The stored procedure was associated with a virtual name that was used in the path for the requesting message.

SQLXML 3.0 also creates a service configuration (.ssc) file when you create the virtual name and associate a stored procedure. The file is created in the path defined for the virtual name. The .ssc file contains information about the stored procedures that act as the methods for your Web service. SQLXML 3.0 uses the information in this file to call your stored procedure correctly. This is critical for the passing of parameters into the stored procedures. When you define the stored procedures for your Web service, you have an opportunity to define the required parameters. This information is saved in the .ssc file and is used to determine the appropriate method for executing the stored procedure.

The process of configuring these files may seem tedious, especially if you have to do it manually. Microsoft provides SOAP Toolkit 2.0, which can read the WSDL file and create an object that supports the methods defined in your WSDL file. You then could create a VBScript that uses the SOAP Toolkit to invoke a method of your newly created Web service. The SOAP Toolkit simplifies the process of interacting with Web services. By creating a SoapClient object through the toolkit, you can invoke the SoapClient object in a manner similar to the way you invoke any other object from VBScript. The client invokes the object and references the URL. The URL includes the following information that you have already defined for your Web service:

- The virtual directory
- The virtual name
- The wsdl parameter (used to point to the WSDL file)

The SoapClient object is created when the client uses the MSSOAPinit method. The SoapClient object then uses the wsdl parameter to request the WSDL file for the Web service from SQLXML 3.0. The SoapClient object parses the WSDL, reviews the content, and then executes the stored procedure as a method of the SoapClient object you invoked.

Thus, the process of invoking the Web service from an application is simplified by SOAP Toolkit 2.0. It is possible to take processes that are defined as stored procedures and make them available through the Web to other Web applications. This maximizes interoperability across platforms and organizations. Web services are a huge part of the future of Web development and may be advantageous by helping you manage your business processes.

Web Services and Enterprise Application Integration

For years many organizations have struggled with the process of creating standards to reuse application logic code. It is desirable to be able to reuse as much of the logic code as possible to decrease the overhead of developing and maintaining applications. It is also important to many organizations for the applications they develop to meet a set of standards.

Traditionally, this has been difficult for several reasons, but one of the major ones is the dependence on a specified vendor. Two of the more common technologies are Microsoft .NET Component technologies and Sun's J2EE platform. These two platforms do not work well together. Each one can be implemented as a Web service, and then the other technology can access the functionality. Web services are built on open industry–standard protocols. This makes them accessible from virtually every development platform. You can create standards that are implemented as business logic and use them for any application that needs the service. Your developers can continue to program in the language they prefer and use Web services to invoke the necessary functionality. Web services give the developers the flexibility they want while maintaining

the standards you want to implement.

For example, say you are a company that sells books. You receive orders via the telephone, the Internet, fax, and mail. You created a Java Web application that accepts orders online by using Java. You also have an application that is used by your customer service representatives to enter all other orders. This application was created by using Visual Basic .NET. Both applications write to SQL Server databases. You then transfer data back and forth between the databases as needed by using the Data Transformation Services (DTS) of SQL Server.

You are frustrated because you have to develop the code for determining shipping charges in both applications but want them to be consistent. You can create a Web service that generates the shipping charge for an order. The Web service then can be used by both applications to assure that both applications use the same logic and achieve consistent results. Then you can be certain that both applications will display the same shipping charges.

If your organization has disparate development environments and you want to implement applications that share a common business logic and standards, Web services can help accomplish this. You should consider Web services to maximize the interoperability of your applications.

Web Services and Business–to–Business Solutions

B2B solutions are implemented so that organizations can share application processes and logic. BizTalk Server is a vital tool in managing B2B solutions. Although BizTalk Server does not use Web services, it does use XML, SOAP, and HTTP. With BizTalk Server a developer can create an environment that is similar to Web services. BizTalk Server does come with a high price tag, and if you require limited functionality and do not have the necessary financial resources or technical expertise, Web services could be used to share information or processes between organizations.

For example, say you are an organization that performs accounting services for several construction companies in your city. You need to provide an interface to your accounting data. Each of the construction companies would like to interface its systems with the financial information you have supplied. You can create a stored procedure in your SQL Server database that exposes the data that are necessary for this reporting. You can create the stored procedure to access the data needed to provide each construction company with the numbers necessary for its general ledger and balance sheet reports. You then can expose the stored procedure as a method of a Web service. From each of the systems at the construction companies, you can interact with the Web service to populate the reports.

If you require continuous transformation and business process services between your organization and others, you should consider BizTalk Server. The features provided with BizTalk Server are superior to those provided with standard Web services. However, if you need only occasional access to logic or data, Web services can provide a solid solution.

Web Services and Business–to–Consumer Solutions

B2C solutions are Web applications that sell products or business services to other individuals or companies. You typically market products at your Web site and allow others to access information and purchase those products. Web services are becoming their own B2C solution. Many development companies have started creating Web services that can be used by other organizations to simplify business processes. A common example of this is credit card authorization. Several development companies have created credit card authorization Web services. A company pays them a fee, typically monthly, and uses the Web service within its Web

application.

If you have a business process that is common or beneficial to other organizations or individuals, you may want to consider selling your business solution as a B2C Web service. While this is not intended to replace your traditional B2C solutions, it can be used to share vital business processes and create revenue.

Summary

Web services are becoming a key component in enterprise development. Web services provide a Web medium to define a uniform standard or service for applications to be based on. In addition, Web services provide an opportunity to market a service that can be integrated into other organizations' applications.

When creating Web services, you should be concerned about five items:

- The standard way in which the data are presented
- The service description language the service uses
- The message format standard
- How the Web service can be found on your Web site
- How to discover service providers for the service

Through XML, SOAP, UDDI, and WSDL standards, you can account for each of these five concerns. Implementing Web services is all about the use of industry standards to expose your service or data.

Web services can be a foundation for your .NET business solutions. Web services can enhance your EAI solutions by providing a Web medium for your organizational rules and standards. Web services can play a role in B2B solutions and B2C solutions as you look at selling or exposing your business process to vendors, trade partners, and other customers.

The .NET Enterprise Servers can take advantage of your Web services by including them within an application or exposing their process as a Web service. A common example of this is exposing an SQL Server stored procedure as a Web service.

Review Questions

1. What is a Web service?
2. How are Web services different from the business layer of the traditional Windows DNA?
3. What is the purpose of WSDL?
4. What is SOAP?
5. What are the necessary steps for exposing an SQL Server stored procedure as a Web service?
6. Why is NLB so important for Web services?
7. What is the role of Web services in B2C solutions?

Chapter 16: Building Enterprise Web Farms

Overview

The purpose of an enterprise Web farm is to provide a Web application that is highly available, secure, and responsive. Most organizations that build Web commerce solutions want the Web applications they deploy to be available for connections all the time. Installing the application on a single server with Windows–based software may not be enough.

The challenge of meeting the goals of a Web farm becomes more difficult as an organization's Web farm grows. Every application is made up of three logical layers that constitute its overall architecture. The layers of a Web farm application, as shown in Figure 16.1, are as follows:

- **Presentation layer.** This is the user interface. For a Web application the interface typically resides at the Web server. For a typical client/ server application the interface is installed on the application user's workstation.
- **Business layer.** The business rules of an application are enforced here. With a Web farm and other three–tier applications the business layer often is installed on a middle server.
- **Data layer.** The data of an application is stored at this layer. In a Web farm and in most other applications this layer resides on its own server. The server is typically a relational database management system. The examples in this chapter use SQL Server.
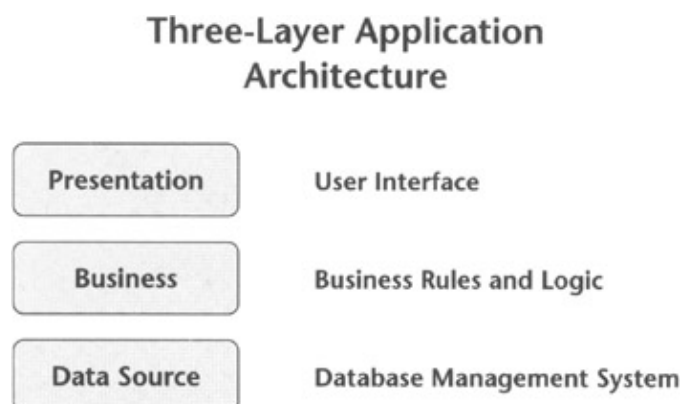
Figure 16.1: All applications have a logical three–tier architecture.

This chapter describes the methods by which an Enterprise Web farm meets its underlying goals. Each of these goals—security, availability, and responsiveness—must be accounted for at each level of the application architecture. Because a Web farm may have multiple servers for each layer of the architecture, it is imperative to understand the requirements of each layer.

For instance, the responsiveness of an application is configured differently at the presentation layer than it is at the data layer because the presentation layer stores no data. The presentation layer facilitates the connection from the user and the retrieval of data from the database. The performance issues are tied to the user connection. The data layer serves the actual data. Improving performance at the data layer includes the use of a different set of variables. At the data layer the user is tied somewhat to the location of the database. It may not be feasible to implement a technology such as load balancing because the data are not stored on all the servers participating in the load–balancing cluster.

This chapter has a section on each layer of an application. Within these individual sections the design goals of the Web farm are addressed. By the end of this chapter the reader will have a solid foundation of knowledge about the design and strategy needed to implement and optimize a Web farm solution.

Presentation Layer

The presentation layer of an application provides two important functions for the application user. This layer provides the user interface for the application. The user interface includes pages, forms, and other objects with which the user interacts directly. The second important function of the presentation layer is interaction with the user. The presentation layer handles the initial connection for the application. This is the application point of entry for the user. This does not mean that the application does not have to connect to other servers and other layers to fulfill the user's requests. Typically, the presentation layer does this on behalf of the user. For a common client/server application the presentation layer is installed on the client machine, often in the form of an executable. The code that makes up the presentation layer has the appropriate information to connect to the database to retrieve and manipulate data.

The architecture changes with a Web application. At this point the presentation layer resides at the Web server in the form of ASP.NET pages or another Web content format. The user connects through the browser to the Web server, which provides the presentation layer. This gives a developer the advantages of decreased maintenance, because the application is stored in one place, and increased exposure. Virtually any user on any machine can use the application because it is accessible through a browser.

This presentation layer becomes an even greater concern as one considers a Web farm solution. A Web farm is a variation of a Web application. With a Web farm it is necessary to add more design considerations to make sure the application scales to meet the current requests, is always available, and is secure. To meet these needs one must implement multiple servers at the presentation layer. The following sections discuss security, availability, and responsiveness options at the presentation layer. Readers also are introduced to the .NET Enterprise Servers that enhance their ability to meet these goals.

Presentation Layer Security

When one is building a highly available Web site, security at the presentation layer should be a concern. Security is most critical at the presentation layer because that layer is the interface with the Web user and is also the piece that is exposed directly to the Internet. There are two primary concerns pertaining to the security of the presentation layer. The first is authentication. An organization wants to make sure that only the users who should have access to the site do. This includes not only the verification of the user's account and password but also the verification of the traffic that is entering the site. Many tools are available to hackers, and you should protect yourself wherever possible.

The second security concern is encryption. After users have authenticated at the site, they will make requests for pages and data. You want to make sure that sensitive data are encrypted to prevent unwanted access to critical information. The following sections provide more detail on options for authentication control and data encryption.

Authentication

Authentication is the process of verifying the credentials of the user who is connecting to a Web site. At the presentation layer the first level of defense is within Internet Information Server (IIS). IIS stores the Web site and the pages that make up the application with which the Web users interact. From IIS there are the following four options for authentication security, as shown in Figure 16.2:

- **Anonymous authentication.** All users are automatically mapped to an anonymous account and are restricted to only the pages the anonymous account can access.
- **Basic authentication.** Users are required to supply a username and a password. The credentials are sent to IIS in a clear text format.
- **Digest authentication.** Digest authentication is relatively new. It passes security credentials through proxy servers in a secure manner. It is, however, dependent on Internet Explorer Version 5.01 or later and Active Directory.
- **Integrated Windows authentication.** One can force the authentication to use the normal Windows authentication methods. This is a secure solution, but it is dependent on the browser. The browser must support Windows authentication (Internet Explorer).
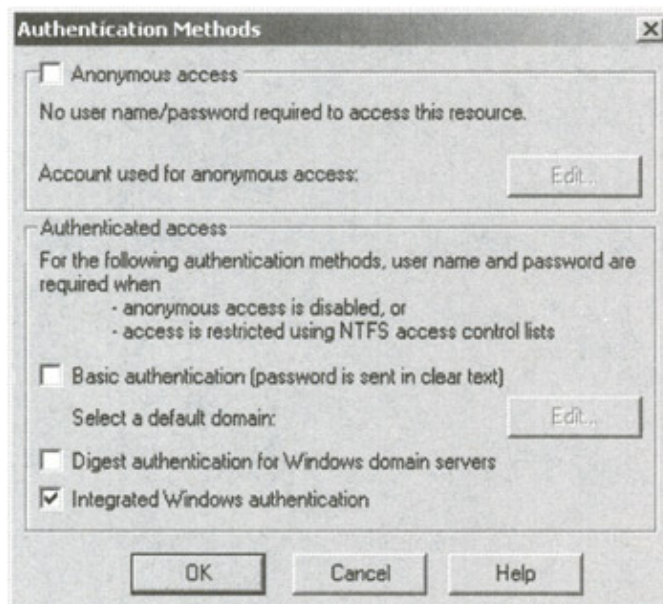
Figure 16.2: From IIS it is possible to configure Integrated Windows authentication to authenticate Web users on the basis of the .NET Active Directory structure.

Authentication options can be expanded further through the use of Commerce Server 2002. By using Commerce Server one can take advantage of the AuthFilter and AuthManager authentication features to customize authentication further. For more information on each of these authentication options, refer to Chapter 10, "Commerce Server."

Encryption

The second area of security concern at the presentation layer is the encryption of the data that are passed over the Internet. When one is implementing a Web farm, the presentation layer is the layer most exposed to the Internet. The presentation layer also is responsible for the interface with the user. The Web user is using a browser to connect to the Web servers that store the Web content that makes up the presentation layer. To maximize the security level it is desirable to implement an encryption technology. The most common encryption technology used on the Internet is Secure Sockets Layer (SSL) security.

To implement SSL one has to obtain a certificate from a known certificate authority (CA). The most common CA is VeriSign. The reader can get more information about obtaining and installing certificates from the VeriSign Web site at http://www.verisign.com/. After you have obtained the certificate and installed the certificate on your Web server, you need to enable SSL for the Web site. The Web site can be enabled through either IIS or Commerce Server Manager.

After the implementation of SSL, the Web site supports HTTPS connections. Those connections are encrypted at either a 40– or a 128–bit encryption level. The connection to the Web server is over port 443 by default.

Presentation Layer Availability

At the presentation layer you typically are not providing data. The application user is interfacing with the ASP.NET or a similar technology that is performing the actual request for data. As far as availability is concerned, you should make sure the connection to the application is always available. Later in this chapter, the section titled *Data Layer Availability* introduces the options for making data highly available. At the presentation layer you want to make sure that a Web server is always available to handle user requests.

The best method for addressing presentation layer availability is through the network load balancing (NLB) service of the .NET Server platform. With NLB it is possible to group multiple servers into a cluster. The cluster, at this layer of the architecture, is just a group of servers. Servers in the cluster alternate requests to share the load related to user connections. If a server goes down, the other servers in the cluster pick up the load and handle the incoming requests. Additionally, you can take a server offline and perform maintenance without affecting connectivity to the site.

With NLB the data are not shared among servers. It is your responsibility to make sure that all the servers have the same pages to handle the client connections. Application Center 2000 can be used to expand the functionality of the NLB service. With Application Center 2000 you can synchronize the presentation–layer pages across all servers.

Presentation Layer Responsiveness

Performance at the presentation layer is certainly a consideration. Because users first interact with the presentation layer, performance issues are most noticeable at this layer. NLB is also a solid method to address any performance issues. Because the servers alternate requests, you can add more servers to the mix without negatively affecting the performance of the existing servers. As the number of hits to the site grows, you can increase the number of servers that belong to the cluster. By using NLB you give the organization the potential to grow as the needs of the site grow. More information on NLB appears in Chapter 9, "Application Center 2000."

Business Layer

The business layer houses the business rules. These rules ensure that the application meets the requirements of the organization and current business processes. These rules also are typically at the core of an application's purpose. This is an opportunity to educate the software on the needs of the business process. In a three–tier application architecture the business layer is controlled by its own servers and is managed by Component Services (COM+).

In a three–tier application model, server components provide centralized multiuser functionality for

an application's client components. These components are managed by COM+. The software that supports this concurrent access to shared business and data services is called the application infrastructure or plumbing. This is software that works with the operating system to provide the centralized multiuser functionality for most data services.

Additionally, COM+ supplies the application infrastructure that enables systems to interoperate efficiently with business logic shared by a large number of users. This reduces both the complexity and the cost of building three−tier applications and enables developers to focus on the application's logic without worrying about the plumbing. This results in less complex, more rapid development. COM+ overcomes the inherent problems of maintaining data integrity and system reliability in shared systems by doing the following:

- Managing processes, system threads, and other operating system resources to facilitate concurrent user access
- Synchronizing access to shared data through connection pooling to maintain performance and conserve server resources
- Performing transaction monitoring to keep each user's actions isolated from the actions of others
- Implementing security so that unauthorized users cannot access the system

COM+, as its own server, is a positive solution for Web farms because it separates the business logic processing from the presentation layer and the data layer. This separation decreases the responsibilities of each of the servers, and this can increase performance as the number of users grows.

The following sections identify the security, availability, and responsiveness concerns related to the business layer and COM+.

Business Layer Security

When the business layer is isolated to its own server, you have to be concerned about how the presentation layer interacts with the business layer server. Then you need to be concerned about the methods with which the business layer interacts with the data layer. You now have a new layer between the application interface and the database and have to account for security at each of these interfaces.

COM+ provides several security features to protect COM+ applications. These services can be configured or implemented either declaratively at design time or programmatically at runtime. .NET Server Component Services provides a rich visual interface to implement security declaratively, while COM+ exposes a rich Application Programming Interface (API) for managing security programmatically.

The automatic security services that COM+ offers (role−based security and authentication) make it possible to implement all security functionality declaratively. When the services are turned on and configured appropriately, COM+ handles the details of enforcing the security of the security policies. Additionally, if the automatic services are not sufficient for the application requirements, it is possible to extend them, building on the automatic security platform provided by COM+.

Which mechanisms are used to secure a particular application depends on the particular requirements of that application. Some security choices affect how you write components, and some can affect the application's design significantly. Before making any decisions about how to secure an application, consider its security requirements in the context of its overall design and

choose the most suitable combination of security features. The following sections describe relevant COM+ security categories and features.

Role−Based Security

Role−based security is the central feature of COM+ application security. By using roles, it is possible to administratively build an authorization policy for an application. Authorizing which users can access which resources can be done at the component level, an interface level, and, if necessary, a method level. Roles also provide a framework for programmatically enforcing security when applications require more granular access control.

Role−based security is built on a general mechanism that enables the developer to retrieve security information regarding the original caller in the chain of calls to a component. This facilitates database auditing and logging when a COM+ application adopts its own security identity.

Client Authentication

Before clients can be authorized to access resources, it is necessary to be able to authenticate who they are. A COM+ application lets you turn on the authentication service administratively to enroll other more fundamental COM+ and .NET Server services. These services are transparent to the COM+ application.

To turn on application authorization checking, it is necessary to enforce access checks for the application. Access checks enable full role−based security. When designing your application, you may want to disable security so that you can focus on the program's design and logic. Complete the following steps to enable access checks for an application:

1. In the console tree of the Component Services administrative tool, right−click the COM+ application for which you want to enable access checks and then click Properties.
2. In the application Properties dialogue box, click the Security tab.
3. As shown in Figure 16.3, select the Enforce access checks for this application checkbox.
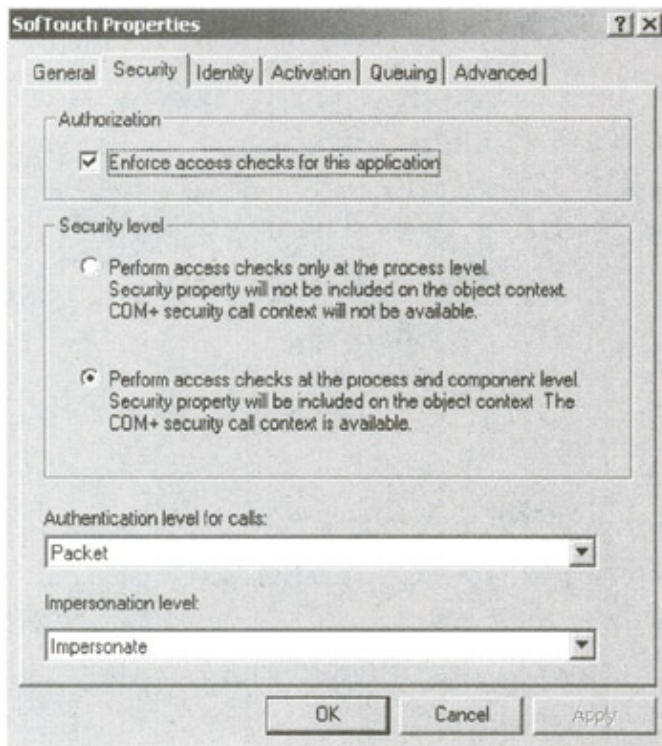
Figure 16.3: It is possible to enforce access checks for a COM+ application to enable role–based security.

    4. Click OK.

After enabling access checks, select the appropriate security level. There are two choices for setting a COM+ application security level:

- **Perform access checks only at the process level.** Security checks are performed only at the application level. Fine–grained role checking is turned off at the component, method, and interface levels. In addition, programmatic role–based security options are unavailable.
- **Perform access checks at the process and component levels.** Security checks are performed at the object, interface, and method levels if so established. Programmatic role–based security also is enabled.

To select a security level, perform the following steps:

1. In the console tree of the Component Services administrative tool, right–click the COM+ application for which you want to enable access checks and then click Properties.
2. In the Application Properties dialogue box, click the Security tab.
3. As shown in Figure 16.3, select either Perform Access Checks Only at the Process Level or Perform Access Checks at the Process and Component Level.
4. Click OK.

Any changes will be effective the next time the COM+ application starts.

Business Layer Availability

After you have chosen to move the business layer to its own server, you have to make sure the business layer is available at all times. You have moved many of the core components of the application to a middle–tier server. If you deploy only a single server at this layer, you create a single point of failure. If the business layer is not available, the application will not function.

You may want to consider implementing a fault−tolerant solution at the business layer. You could implement at least two servers and provide the components at both of the servers. The NLB service of .NET Server does not load balance components. To make this work you need to deploy Application Center 2000 on the middle−tier servers. Application Center 2000 includes a component load−balancing feature that allows you to take advantage of both servers in the middle tier. You also can take one of the servers offline either intentionally or unintentionally without affecting the connectivity of the application. By implementing multiple servers at the business layer and deploying component load balancing, you increase the availability of the Web farm.

Business Layer Responsiveness

The business layer is a key factor in the overall performance of a Web farm. The layered architecture of an application is logical. All three layers of the application could reside on the same machine. Of course, as the number of users increases, it becomes necessary to consider offloading some of the application processing.

The performance consideration is the server hardware. One needs to decide whether the business layer justifies its own server or cluster of servers. The additional servers increase the hardware and software overhead for the application but also offload some of the processing of the Web farm. If the primary concern is the availability and responsiveness of the application, you should consider a cluster of servers. You have to compare the price of additional servers and the licensing of .NET Server with the advantages of availability and responsiveness.

You can move many of the core application components to the middle tier to decrease the overhead of the presentation layer. The presentation layer then will be able to handle more connection requests. The business logic of the application also can be moved to the middle tier. This will offload some of the processing of the data layer. The data layer then will be able to expend more of its resources in data retrieval.

If you have chosen to implement multiple servers at the middle tier, you need to deploy Application Center 2000. You can use the component load−balancing service to alternate requests of the middle layer between the servers that have been deployed. This is an effective way to distribute the load of the requests of a middle tier COM+ server.

Data Layer

The data layer is primarily responsible for storing data. When the data have to be retrieved, this is done through a connection to a database. The database generally is stored in a relational database management system. In this book and in the examples SQL Server is used as the database management system for the databases used in a Web farm.

From the perspective of the data layer the most important goal of a Web farm is availability. Security and performance are also concerns and will be addressed in more detail in the next couple of sections. At the presentation layer you were concerned about the availability of the connection. At the data level you should be concerned about the availability of the data. The presentation layer supports the connection to the user and then connects to the database on behalf of the application user to access the data to fulfill the user's request. The following sections outline the considerations of security, availability, and responsiveness at the data layer.

Data Layer Security

When the presentation and business layers need to connect to the data layer to service the request, a connection is made to the database. SQL Server supports two types of authentication for its connections: Windows authentication and SQL Server authentication:

- **Windows authentication.** Windows authentication integrates Windows 2000 security with SQL Server security. With this authentication option the client can use the same account for logging into Windows 2000 and SQL Server 2000. Windows authentication is dependent on the user first logging in successfully to the Windows 2000 domain. A user connecting to SQL Server by using Windows authentication is referred to as a trusted connection. From the presentation or business layer you need to supply the connection information to connect via Windows authentication. The following code is an example of performing a Windows authentication connection. The actual syntax may change for different programming languages. The user should consult the product documentation to find the code for performing a Windows authentication connection to an SQL Server database. The pieces of code in boldface represent the area of the example that identifies the type of connection being made:

```
dim cn
set cn = Server.CreateObject("ADODB.Connection") 'creates connection
object
cn.Provider = "SQLOLEDB"  ' native SQL Server provider identifier
cn.ConnectionString =       "Data Source=ServerName;" & _
    "Initial Catalog=DatabaseName; " & _
"Integrated Security-SSPI;"
cn.Open
```

- **SQL Server authentication.** When a Windows 2000 domain has not authenticated the user, the only other option for accessing SQL Server is through an SQL security account. The user can supply user credentials to the SQL Server, bypassing Windows 2000 security. After the user passes login credentials to the server, SQL Server compares the credentials submitted against the list of SQL Server 2000 security accounts. If the SQL Server finds the credentials in the sysxlogins table, the user is granted access to SQL Server. When one is using SQL Server authentication, the user name and password are supplied to a server in clear text format unless both the client and the server are using SSL encryption for the entire session. The following code is an example of a connection made to SQL Server by using SQL Server authentication:

```
dim cn
set cn = Server.CreateObject("ADODB.Connection") 'creates connection
object
cn.Provider = "SQLOLEDB"  'native SQL Server provider identifier
cn.ConnectionString =       "Data Source=ServerName; " & _
    "Initial Catalog=DatabaseName;" & _
    "User ID=UserName;Password=UserPassword;"
cn.Open
```

Data Layer Availability

SQL Server is the back end to the site. In addition to being used for the data that need to be presented to users, it houses the databases used for Commerce Server and BizTalk Server (if it is in use).

You will want to make sure to take all the necessary precautions with the database. In addition to making normal backups, you want to take some additional steps to ensure availability. In fact, you

want to take steps to make sure you never will need to use the backups. SQL Server provides three strategies for creating highly available databases:

- Clustering
- Replication
- Standby server (log shipping)

The following sections describe each of these options in more detail. For a detailed description of the configuration of each of the following options, refer to the SQL Server Books Online.

Clustering

As shown in Figure 16.4, two or more SQL Servers sharing common data form a cluster of servers that can work together as a single server. Each server is called a node and can operate independently of the other nodes in the cluster.

Two-Node Cluster Configuration
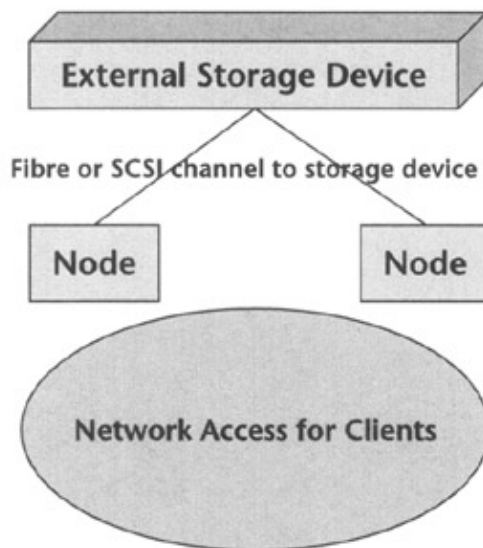
Figure 16.4: Clustering can provide fault–tolerant support for SQL Server instances.
Each node has its own memory, system disk, operating system, and subset of the cluster's resources. If one node fails, another node takes ownership of the failed node's resources. The cluster service then registers the network address for the resource on the new node so that client traffic is routed to the new server. When the failed server is brought back online, the cluster service can be configured to redistribute resources and client requests.

SQL Server can be configured in either a two– or a four–node cluster and can be configured as an active/active cluster or an active/passive cluster. In an active/active cluster both servers are used while in regular operation. When one of the servers fails, the other simply picks up the additional responsibilities. In an active/passive mode one server acts as a warm backup: It is ready to go if the primary server fails. There is more information about clustering in Chapter 18, "Clustering .NET Servers."

Replication

SQL Server replication can be used to synchronize a site's database servers. SQL Server replication options include snapshot, transactional, and merge replication. You should consider

using SQL Server replication for batch updates if your application logic is aware of the nature of database access. Replication is a cheaper solution than clustering is. It is also more difficult to configure, and the application logic concerning connections requires more maintenance.

Standby Server

If a single production server provides both read and write access to data, you should log all transactions. You can use SQL Server log shipping to transfer files to a nonproduction server that is updated continuously with the transaction log files. This technique is called warm backup because the backup database is always "warmed up" by continuous application of the transaction logs. This option is inexpensive and easy to manage and provides a strategy for availability in environments where there is some tolerance for downtime.

This method can be used in combination with the strategies described in the preceding sections to recover from disaster when availability is particularly critical. You also can use warm backups to keep geographically dispersed data centers reasonably synchronized. In fact, this is an inexpensive way to maintain a synchronized database in a separate data center for site disaster recovery.

Data Layer Responsiveness

In addition to having the data available, you should be concerned about the performance of data retrieval. As the number of users at the Web farm grows, you want the database retrieval to continue to perform quickly.

To optimize responsiveness at the data layer, consider the normal optimization features available in SQL Server. You will have to verify the configuration of each of the following items. Each item is a factor in the overall performance of the database:

- Indexing strategy
- Memory configuration
- Processor configuration
- Denormalization techniques
- Archiving strategies

This list does not include all the factors that affect performance at the data layer but is a good starting point. For further information about the configuration and optimization of each of these areas and about tuning SQL Server databases, refer to the *SQL Server Books Online.*

Summary

A Web farm, like other applications, has a three–layer architecture. In most Web farm scenarios each layer is deployed on at least one server, and in many cases each layer has multiple servers. To deploy a Web Farm effectively, you should strive to make each of these layers secure, available, and responsive.

You can use the .NET Enterprise Servers to build an Enterprise Web farm. IIS, Application Center 2000, and Commerce Server can be effective tools in configuring the presentation layer. Application Center 2000 increases the availability and responsiveness of the presentation layer by providing load–balancing services. Commerce Server is an effective tool to enhance the security of the presentation layer. COM+ and Application Center 2000 can be used to deploy the business layer of an application to its own group of servers. Application Center 2000 provides a component

load–balancing service that increases the availability and responsiveness of the middle tier. COM+ has several built–in security options to help manage the business layer. At the data layer SQL Server is the key product. The data are stored in an SQL Server database. You can enhance the availability of Web farm data further by implementing the cluster service of the .NET Server platform.

An Enterprise Web farm is a strategy that is deployed across multiple servers and uses many of the .NET Enterprise Servers. You should design your strategy and be aware of the number of servers and products you require before starting to implement the solution. By using the .NET Enterprise suite of products you can create a secure, available, and fast Web farm solution that will grow as the business's needs for a Web farm grow.

Review Questions

1. What are the goals of an enterprise Web farm?
2. What is the role of the presentation layer?
3. What are the primary security concerns at the presentation layer?
4. What can be done at the presentation layer to increase availability?
5. What is the role of the business layer?
6. What is component load balancing? Why would a developer consider using it?
7. What is the role of the data layer?
8. What can be done to ensure availability of data?
9. What types of authentication does SQL Server support?

Chapter 17: Data Warehousing

Overview

Data warehousing is a technology that often is misunderstood and used as a general term. A data warehouse is a database that is structured for querying and analysis. The data typically are organized in a manner that represents the business history of an organization. The data generally are summarized and read only. The following characteristics describe a data warehouse.

- Data are transferred from other locations.
- Data are made consistent before their storage in the data warehouse.
- Data are summarized.
- Data are stored in a format that is ready and convenient for querying and analysis.

A data warehouse is created as a database in SQL Server 2000 or another database management system. For the purposes of this book, SQL Server 2000 is used as the data warehouse database.

Analysis Server is used in conjunction with SQL Server 2000. Technically, it is not part of SQL Server; it is a separate product. Analysis Server extends the normal relational functionality of SQL Server to optimize the analytical nature of a data warehouse. The purpose of the product is to give the end user a fast, easy–to–use, and intuitive look at the data needed for analysis.

Data warehousing has been around for years. Many of the design concepts provided in this book are not new technologies. The exciting features that are being implemented with the new technologies involve the Internet. Many organizations are evaluating methods to make analysis data available in a Web forum. This is done not only to allow access to the data by Internet users but also to allow internal users to have a thin–client solution; this can be beneficial for licensing and support.

The focus of this book is on introducing solutions that can be deployed with .NET Enterprise Servers. A data warehouse is one of these solutions, but it is necessary to add on to the normal implementation and describe the solutions available to the user for integration with the Internet. The details in this book are focused on design and general implementation information. The reader will be introduced to design strategies for a data warehouse and for integrating an organization's solution with the Internet.

The specifics of the configuration within Analysis Server are not described. Those who are just starting out with SQL Server Analysis Services should refer to *Microsoft SQL Server 2000 Analysis Services Step by Step* published by Microsoft Press. This book will get you going in the right direction with SQL Server Analysis Services.

This chapter introduces the data warehouse framework. This portion of the chapter defines the components and tools that make up the framework of the data warehouse. The chapter then describes Analysis Server, which is the core tool for administering and configuring the data warehouse and online analytical processing (OLAP) databases. The chapter then compares data warehousing with OLAP.

These terms often are used interchangeably but actually refer to two separate layers in the Analysis Server architecture. It is important to understand the difference between the two because they often are installed on different servers. In the OLAP section the reader is introduced to the cube. The cube is the OLAP component that is used for interaction with the data. For users, a cube is a

retrieval point for data access.

The chapter concludes with a brief description of the data access options for OLAP databases. In the data access sections the reader is introduced to the options available for accessing the data warehouse with the .NET services. This includes a section on interaction with the other .NET Enterprise Servers.

Data Warehouse Framework

The Data Warehouse Framework is a set of components, tools, and services that provide the foundation for an implementation of data warehousing. Building a data warehouse is a process of design, implementation, and maintenance. There is no single service that provides all these features. In SQL Server 2000 it is desirable to implement the following list of services and tools to implement a data warehouse:

- SQL Server databases
- Data Transformation Services (DTS)
- Analysis Services
- PivotTable Service

The following sections describe the role of each of these components. The flow of data from the transactional processing system to the user that is making the request is dependent on the services mentioned above. Figure 17.1 identifies the flow of data from the transactional processing environment to the client that is performing the data analysis. The data are entered into the transactional processing databases. From that point DTS can be used to transfer the data to the database that is being used for the data warehouse. This process may involve transforming the data to the format and summary level that is useful for the data warehouse. The data then are moved from the data warehouse database to the cube. The cube is an object within an Analysis Services database. The process of copying the data to the cube also can be performed by using DTS. This is referred to as processing the cube. At that point the clients can access the data from the cube. The client makes the request from an application that uses the PivotTable Service to access the data. The PivotTable Service is an Object Linking and Embedding Database (OLE DB).
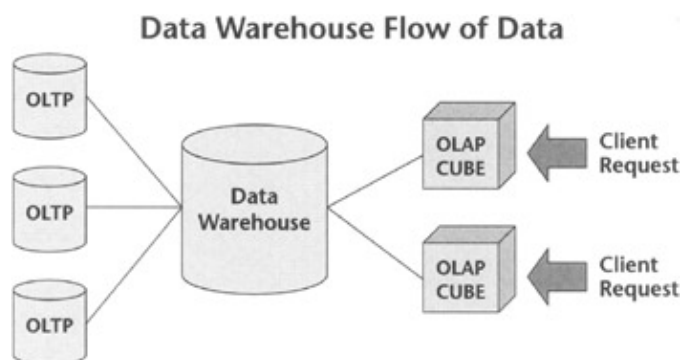
Figure 17.1: The data warehouse framework provides for the flow of data from the transactional processing system to the end user that needs to analyze the data.

Because the PivotTable Service is an OLE DB provider, there are several options for interacting with the data. The data can be accessed from a local client, a network client, or Internet users. This provides the options needed to make the data warehouse accessible enterprisewide.

For example, the user could deploy a Web application that contains Active Server Pages (ASP.NET pages). Within the code of the ASP.NET pages you provide the connection options to connect to the

cube data. In this model, the Internet user connects to the Web server, and the Web server connects to the Analysis Server cube. You can take the architecture and make your cube data available to users regardless of their location. This implementation facilitates analysis and reporting from Web interfaces that previously were not readily available.

SQL Server Databases

The transactional processing databases and the analytical databases are generally SQL Server databases. Both systems could be databases from another product. If disparaging database management systems are used, the data transformation processes are increased and can be more complex.

Data Transformation Services

DTS moves data from one location to another. During this process DTS has the ability to transform and format the data to comply with the needs of the analytical environment. Often the data that are being moved from the transactional processing environment to the analytical processing system have to be transformed or changed. The following list explains why data may have to be transformed:

- Different data types are used on the transactional and analytical databases.
- The data need to be made consistent.
- The data in the transactional environment are not formatted the way the developer would like to analyze them.

With respect to data warehousing, DTS plays a large role in processing the data flow. DTS generally is used to copy the data from the transactional database to the analytical database. DTS is often the catalyst for performing the data transformations required to get the data into the analytical system in a manner that is easily accessible. DTS also can be used to automate the processing of cubes. Cubes and their processing are described in more detail later in the chapter in the section titled *Data Warehousing and OLAP*.

Analysis Services

Analysis services are made up of two main components: Microsoft OLAP databases and the data mining tool. Analysis Services are described in more detail in the section titled *Analysis Server*.

PivotTable Service

The PivotTable Service is an in–process desktop OLAP server that interacts with Analysis Server and provides interfaces for use by client applications that need access to OLAP data on the server. The PivotTable Service is an OLE DB for the OLAP provider. The PivotTable Service provides online and offline data analysis functionality. Examples of using the PivotTable Service are provided later in this chapter. The PivotTable Service is used to connect to the data and perform the analysis. The PivotTable Service can be used from Microsoft Office products, a vendor–created application such as Brio or Proclarity, or a Web interface using Office Web Components (OWC) or the thin client for Analysis Server. Each of these data access options is described in more detail in the section *Data Retrieval from a Cube*.

Analysis Server

Analysis Server is the service component of SQL Server 2000 Analysis Services. It is designed specifically to create and maintain multidimensional data structures and provide multidimensional data in response to client queries. A multidimensional data structure is one that efficiently stores and retrieves data that are stored on multiple axes. A relational format stores all data in a system based on columns and rows, and all data retrieval is based on those columns and rows. In a multidimensional format it is possible to store a measure, a quantitative and additive value such as total price, by multiple factors that replace the standard columns and rows. For instance, if a user was tracking total sales, the user could retrieve on the basis of any number of criteria, such as customer, location, and time. Analysis Server allows a user to store data natively in this format to decrease data duplication and increase access time. Analysis Server is a means of separating transactional data from analytical data to decrease the contention for data access. This also can be used as a security strategy. If there are users who need access to the data solely for analytical purposes, they no longer need access to the transactional processing system. Similarly, if there are users who do not need access to the analysis data, it is not necessary to give them access to the OLAP databases.

Analysis services are made up of two main components: Microsoft OLAP databases and the data mining tool. The following sections describe the role of each of these components.

OLAP Databases

OLAP databases provide an analytical alternative to the standard relational database architecture. The purpose of designing an OLAP database is to provide fast query response time, flexible viewing options, and multidimensional analysis. These are the characteristics of an OLAP database:

- OLAP databases are denormalized to facilitate fast user queries.
- OLAP databases have a robust engine to provide fast numeric analysis.
- OLAP databases are intuitive to users.
- OLAP databases are separate from the transactional processing system. This can provide an additional layer of security.

OLAP databases are multidimensional. They use cubes to allow the viewing of numeric data summarized by varying characteristics. Cubes are described in greater detail later in this chapter.

OLAP databases are excellent for applications that revolve around an analytical process. The following applications are good candidates for OLAP databases:

- Financial reporting applications
- Sales/marketing analysis applications
- Executive information systems
- Statistical analysis packages
- Any application that requires heavy calculations for analytical purposes

Data Mining

Data mining is the process of finding meaningful patterns in large amounts of data. Data mining models can be configured to search for the patterns within data. This information can be used for historical analysis and the prediction of future outcomes. One starts with data mining by creating a data mining model.

A data mining model is a virtual structure that represents the grouping and predictive analysis of relational or multidimensional data. In many ways the structure of a data mining model resembles the structure of a database table. However, while a database table represents a collection of records, or a record set, a data mining model represents an interpretation of records as rules and patterns composed of statistical information, referred to as cases. The structure of the data mining model represents the case set that defines the data mining model, while the data stored represent the rules and patterns learned from processing case data.

The data mining model of SQL Server is easy to use. The data are analyzed by the data mining discovery process, and users do not have to perform all the analysis. This allows the users to get a more meaningful view of their data. The data are shown to them in a result format, and they do not have to have full access to all the data that the data mining model has analyzed.

Data Warehousing and OLAP

Although they sometimes are used interchangeably, the terms data warehousing and OLAP apply to different components of systems that often are referred to as decision support systems or business intelligence systems. Components of these types of systems include databases and applications that provide the tools analysts need to support organizational decision–making.

A data warehouse is a database containing data that usually represent the business history of an organization. These historical data are used for analysis that supports business decisions at many levels, from strategic planning to the performance evaluation of a discrete organizational unit. Data in a data warehouse are organized to support analysis rather than to process real–time transactions as they are organized in online transaction processing systems (OLTPs).

The data warehouse is the underlying data structure for the OLAP database. The OLAP database consists primarily of cubes. A cube is a multidimensional data structure that is accessed for data analysis. The data warehouse is a typical database in a relational database management system. For the purposes of this book it will be assumed that the data warehouse is stored in SQL Server. The cube is stored in a multidimensional format on Analysis Server. The data from the data warehouse are populated to the cube through a step referred to as processing the cube. The analytical users interface with the cube, and the cube is based on the data warehouse.

For example, you could deploy a data warehouse with the following server strategy. You have several online transactional systems from which you transfer summary data to create your data warehouse. These data are transferred to your SQL Server data warehouse through the use of DTS. You then could create an Analysis Server OLAP database that is dependent on the SQL Server data warehouse. Although it is possible for the data warehouse and the OLAP database to be on the same machine, this is not a requirement. The users can then access the OLAP data for analysis from Excel on their local machines.

The following sections describe the design of this architecture. The first section outlines the core of the data warehouse design; then the OLAP database and the cube are introduced.

Data Warehouse Design

Before one can create an OLAP database and define cubes to navigate through data, one must first supply the source for the OLAP database. The source is generally a data warehouse. The key to making an OLAP database efficient is the design of the data warehouse. Although a data warehouse is referred to as a single entity, it is common to have multiple fact tables for a single data

warehouse and multiple data warehouses for the entire organization. Even though these data warehouses are managed separately, they commonly are referred to as a single entity.

If it is not configured correctly, Analysis Services will never work correctly or efficiently. This book focuses on security, and the design topics presented here are for introductory purposes only. For more information on designing a data warehouse, read *The Data Warehouse Lifecycle Toolkit: Tools and Techniques for Designing, Developing, and Deploying Data Marts and Data Warehouse* published by John Wiley & Sons.

A data warehouse should have the following characteristics:

- The data should be organized around subject areas. These areas should match the needs or structure of the organization. For example, you may have a sales data warehouse and a legal analysis data warehouse that use different data. Often these smaller subsets of data that are organized by subject areas are referred to as data marts. The data from the data marts can be rolled up (summarized) into a single data warehouse.
- The data should be nonvolatile and mass loaded. It is normal for a data warehouse to be truncated and reloaded every week or every evening. The analysis expectations should match the requirements of the database load.
- The data should be in a format that is easily understood. Many data warehouse clients provide direct access to the data, as the data are formatted, in a data warehouse. One should avoid meaningless codes for descriptive fields. A meaningless code is a number that represents a value, and the representation is not intuitive. For example, a region could be displayed to a user as North instead of using a meaningless code such as RN. Codes are appropriate in a data warehouse only when they are the common terms used by the end user.
- The data should provide the means for analysis at multiple levels of granularity. For instance, if you are tracking the sales of your organization, you may want to track the date and time of a transaction. The data warehouse should provide means for users to analyze data at the year, quarter, month, and day levels.

Let us now examine the details of the design process. This section will introduce the star schema, describe the fact table and its components, and describe the dimension table characteristics.

Star Schema

The data warehouse typically is structured into what is referred to as a star schema. A star schema includes the fact table and multiple dimension tables. OLAP cubes use the star schema as the data source. Figure 17.2 shows the star schema. The following characteristics help define the star schema structure:

- The center of the star is the fact table, which keeps the instances of the action that are being tracked. For instance, if you were tracking sales entries to the invoice level, the fact table would have one record for each invoice.
- The points of the star are the dimension tables. These tables are used to describe the information that is stored in the fact table. For this example, it is desirable to have a dimension table for each of the details of an invoice. This could include a dimension table for each of the following: customers, inventory, time, and location.
- When one creates a dimension table that has another table joined to it that is not the fact table, one has created a snowflake schema. This is referred to as a snowflake design because there are multiple steps to get to the fact table and because each design starts to look different from the others. The details about the action being monitored may now be two

or more tables away from the fact table. This increases the number of joins necessary to retrieve data and probably will slow performance.
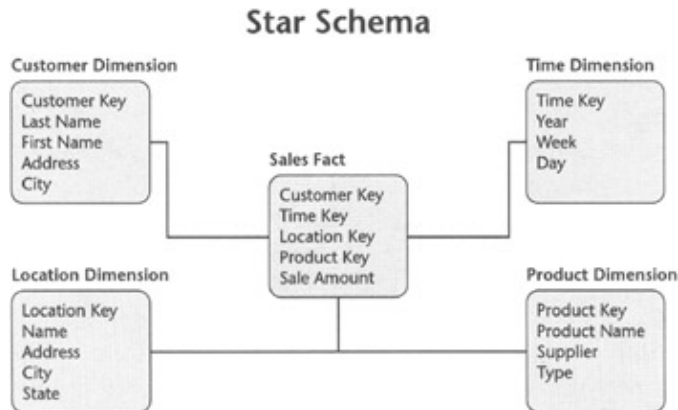
Star Schema

Figure 17.2: The star schema is the preferred method for data warehouse design. The dimension tables are all related directly to the fact table, and this reduces the number of joins necessary for data retrieval.

Fact Table

Each data warehouse or data mart includes one or more fact tables. Central to a star or snowflake schema, a fact table captures the data that measure the organization's business operations. A fact table may contain business sales events such as cash register transactions or the contributions and expenditures of a nonprofit organization. Fact tables usually contain large numbers of rows, sometimes in the hundreds of millions of records when they contain one or more years of history for a large organization.

A key characteristic of a fact table is that it contains numerical data that can be summarized to provide information about the history of the organization. Each row in the fact table has columns that are foreign keys to each of the dimension table's primary keys. The fact table also includes a multipart index that contains as foreign keys the primary keys of related dimension tables that contain the attributes of the fact records. Fact tables should not contain descriptive information or any data other than the numerical measurement fields and the index fields that relate the facts to corresponding entries in the dimension tables.

The most useful measures to include in a fact table are numbers that are additive. Additive measures allow the user to obtain summary information by adding various quantities of the measure, such as the sales of a specific item at a group of stores in a particular time period.

In designing a fact table, the most important step is to identify the granularity of the table. Microsoft most commonly refers to the granularity as the fact table grain. The grain is not a configuration option; it is a statement of the level of detail that the developer wants to track. You must be familiar with the users' analysis needs to choose an appropriate grain. For instance, if you own a retail store, you have the option of storing data by an invoice or by a line item within the invoice. You should have many more line items than invoices. If you choose to define the grain at the line item level, you should be prepared to use the additional storage. You should store data only at the level that is appropriate for user analysis. If there is no analysis at the line item level, do not store data to that level of detail. In choosing the grain for a fact table, consider these items:

- You will not be able to analyze data in any greater detail than your grain. If you choose the grain at the invoice level, you will not have access to the line item information.
- The measures should be consistent with the grain. Your comparative values should identify the grain (fact) of the fact table. For instance, you should not store an invoice total price in a

246

fact table that is tracking information at the line item level. It would be more appropriate to define a measure specific to the line item. This could include quantity or unit price.

- The fact table grain is the most important decision that will be made pertaining to the amount of hard drive space the data warehouse absorbs. Take care in choosing the level of data you need to track. The last level of detail is generally the level that adds the most data. For example, if you chose to store data at the day–by–day level instead of the monthly level, you will increase the number of members of the bottom level from 12 (months) to 365 (days) per year.

Dimension Tables

Dimension tables surround the fact table. They are the points of the star schema created with the data warehouse design. They are descriptive in nature and provide the details of each fact described by the fact table.

For example, if you were creating a data warehouse that was tracking the sales of your products, the fact table would keep a row for each of the sales. The fact table would have foreign key relationships to the dimension tables that provided the descriptions for each of the sales. Dimension tables provide the details of each sale. Some common dimension tables for a sale would provide the following details: the customer, the time of the sale, the location of the sale, the salesperson who sold the product, and the product that was sold. Each of these details would be stored in its own dimension table. The fact table simply combines the details for the specific sale that took place: This customer purchased this product from this salesperson at this time. The following list of characteristics defines the use of dimension tables. Dimension tables are used as the source for cube dimensions:

- Dimension tables provide the context or description of the information stored in the fact table.
- Dimension tables typically have far fewer records than does the fact table.
- Dimension tables create hierarchies. For instance, a time dimension table would have separate columns to represent the year, quarter, month, and day data.

Introduction to the OLAP Database

The OLAP database is the component of Analysis Services that is based on the data warehouse. Each OLAP database references at least one fact table. You can create cubes based on the fact and dimension tables of the underlying data warehouse database.

The cube is the very core of OLAP technology. A cube is a set of data that is a subset or summarization of the data source. In most cases this data source comes from the data warehouse that has been created. The data source for a single cube is generally a single fact table from the source data warehouse. Cubes are made up of dimensions and measures. Although dimensions and measures are objects that are different from the dimension tables and measures that were defined in the data warehouse, they generally correlate with each other. Dimensions and measures will be defined further in the next section. This section provides an overview of the cube structure. A cube is a data structure that is easily interfaced from a client application. The cube structure is multidimensional in nature. It often is compared to a spreadsheet PivotTable. Client applications interface with the cube to view the data they need to analyze. The client applications use the PivotTable Service to interact with the OLAP databases. The PivotTable Service can be used with Microsoft Office 2000 or later versions. It also can be implemented from the PivotTable control, which is an ActiveX control. This control allows Web access to OLAP databases.

Excel and the Web Pivot control are examples of Microsoft clients that can interface with cubes. These front–end applications spare the user from having to write language–based queries. Additionally, users who have Analysis Manager installed on their machines can use the Cube Browser of Analysis Manager to view the data. The cubes store summary information and precalculated data. The data can be navigated easily, similar to a PivotTable, as shown in Figure 17.3.
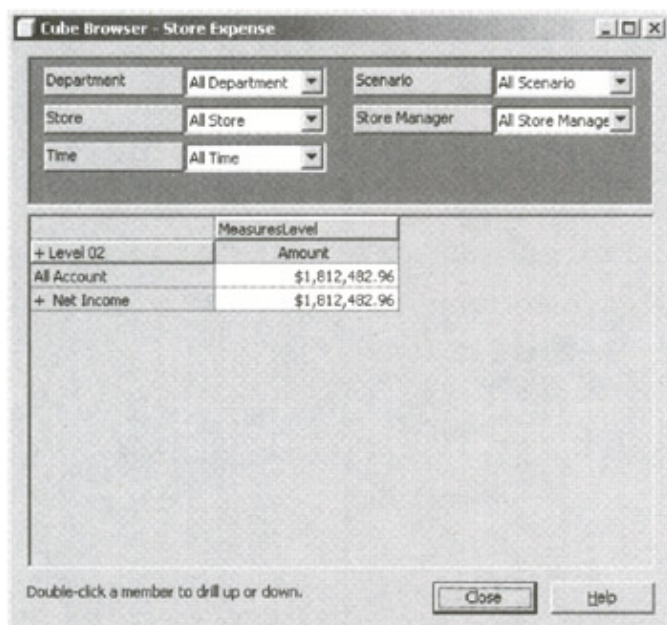
Figure 17.3: The Cube Browser is a tool that makes it easy to browse through cube data. Drop–down boxes can be used to drill up and down through the dimensions of the cube.

The precalculated data are referred to as an aggregation. Aggregations are saved and can be retrieved easily and quickly by users. The aggregations are created for a cube before end users access it. Analysis Server can support many different cubes, such as cubes for sales, inventory, and customers.

Every cube has a schema, which is the set of joined tables in the data warehouse from which the cube draws its source data. The central table in the schema is the fact table, which is the source from which the cube draws its measures. The other tables are dimension tables, the sources of the cube's dimensions.

A cube is defined by the measures and dimensions it contains. Measures are comparative additive values that are used for the detailed analysis. The dimensions are the descriptive information about the measures. For example, a cube for sales analysis may include the measures Item_Sale_Price and Item_Cost and the dimensions Store_Location, Product_Line, and Fiscal_Year. This cube enables users to separate Item_Sale_Price and Item_Cost into various categories by Store_Location, Product_Line, and Fiscal_Year.

Each cube dimension can contain a hierarchy of levels to specify the categorical breakdown that is available to users. For example, a Store_Location dimension may include the level hierarchy: Continent, Country, Region, State_Province, City, and Store_Number. Each level in a dimension is of finer granularity than its parent is. For example, continents contain countries, and states and provinces contain cities. Similarly, the hierarchy of the Fiscal_Year dimension includes the levels Year, Quarter, Month, and Day. When a user is moving to a more detailed level, the process is referred to as drilling down. When the user accesses a level that is less detailed or higher up in the hierarchy, the process is referred to as drilling up.

Dimension levels are a powerful data modeling tool because they allow users to ask questions at a high level and then expand a dimension hierarchy to reveal more detail. For example, a user starts by asking to see Item_Cost values of products for the last three fiscal years. The end user may notice that 1998 Item_Cost values are higher than those in other years. Expanding the Fiscal_Year dimension to the month level, the user sees that Item_Cost values were especially high in the months of January and August. The user then may explore levels of the Store_Location dimension to see whether a particular region contributed significantly to the high Item_Cost values or may expand into the Product_Line dimension to see whether Item_Cost values were high for a particular product group or product. For more information about defining different types of dimensions and measures, refer to *SQL Server Books Online*.

Data Retrieval from a Cube

Data access from the data warehouse is the key to the ability of the developer to provide the end user with the data that need to be viewed. Several tools help with the access of data from OLAP cubes. This section introduces the key data analysis concepts. It then describes the PivotTable Service in more detail and identifies the core options for accessing the data and presenting the data to the user.

Data Analysis Concepts

Analysis Services allows direct access to the data that are stored in cubes. This is done through the Analysis Manager and through the services provided with Office 2000 and later versions. The cubes also can be accessed programmatically. As the data are accessed, knowledge of the meaning of a few key terms can help the user understand the type of analysis made possible through Analysis Server:

- **Drilling down.** Drilling down describes a request for more details. Typically, the user is viewing the data at a higher level than is required and needs to get more information. For instance, if you were analyzing the total sales of your product by location, the initial view of the data would be summarized for all locations. If you wanted to see the details for each state, you would request, generally by double–clicking, more detailed information.
- **Drilling up.** Drilling up is exactly the opposite of drilling down. The user is looking a data that are too detailed and needs to see the data in a more summarized format.
- **Drilling across.** It may be desirable to compare data for two different business processes. Drilling across is an analytical process that compares the data of two different facts. Those facts then can be analyzed against each other, for instance, in the case of a video store that was storing data about the sales of videos separately from the sales for video games. Drilling across would allow the store to compare the details of those two separate facts.
- **Drilling through.** Sometimes an organization may need to query data that are more detailed than the information that is stored in the data warehouse. Drilling through is the process of requesting a detail that has to go back to the source of the data to return the information. Drilling through allows the user to make a request that goes through the data warehouse and back to the transactional processing system. Because the data warehouse consists of summarized data, it may not contain the details of each individual transaction. By drilling through, it is possible to access the more detailed data stored in the transactional processing database.

PivotTable Service

The PivotTable Service is an OLE DB provider that supports OLE DB for OLAP and provides multidimensional extensions (MDXs). The architecture of the PivotTable Service is shown in Figure 17.4.
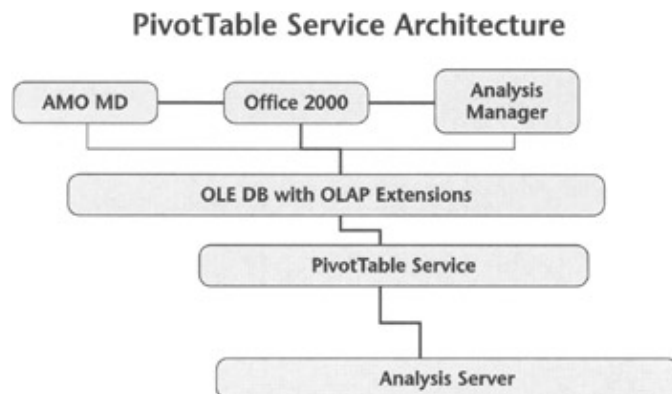
Figure 17.4: The PivotTable Service is the key component in allowing client access to the Analysis Server and cube data.

As the figure demonstrates, client access to the Analysis Server is funneled through the PivotTable Service. The PivotTable Service is a client–side component that allows multidimensional data to be used in Microsoft Office products and custom applications. The PivotTable Service provides the following features for accessing data:

- Allows smart caching of commonly used data
- Connects to the Analysis Server on behalf of the client
- Provides an interface to cube data
- Allows cubes to be downloaded and stored locally
- Offers access to relational data as well as multidimensional data

The PivotTable Service is used by applications that are referencing data from the cube. The following section describes the different options for implementing the PivotTable Service. As a developer uses the applications and interfaces described in the section titled *Options for Accessing Data*, he or she is using the PivotTable Service and therefore has access to its features.

Options for Accessing Data

A developer wants to allow the application and the users to interface with the data warehouse data. Through the PivotTable Service a few options exist for accessing the data stored in an organization's cubes. The following list identifies the options for accessing the data stored in a data warehouse:

- **Analysis Manager.** Analysis Manager is the tool that ships with Analysis Server and is used to view and manage data in Analysis Server. This is a great tool for manipulating the design of OLAP data and can be used to define data mining models. It is probably not the best solution for the average user because it gives the user access to more features than the developer probably feels comfortable with.
- **Microsoft Office products.** Beginning with Office 2000, specifically Excel, can be a great tool for analyzing data. It is possible to use Microsoft Office products to connect to the data in OLAP cubes, generate local cube data (download a portion of the data to a local spreadsheet), and filter the data to view only the pieces that are valuable to the user's analysis. This is a great option for power users but may be too much for users who are not

comfortable working with the advanced features of Office 2000.

- **ActiveX Data Objects MultiDimensional (ADO MD).** ADO MD allows an application to access the PivotTable Service through OLE DB. Through this interface it is possible to create a custom application that retrieves the data necessary for an application to function normally. This extends the normal ADO.NET functionality and allows Visual Basic .NET or another language to be used in developing applications that interface with cube data.

  Note The information on the ADO MD object model is not available in the SQL Server documentation. It can be retrieved from the *ADO MD Programmers Reference*, which is included with ADO 2.x and later.

- **Web Interfaces.** It also is possible to access cube data from a Web interface by using the options described in the next section. This allows for the deployment of thin clients.

Interaction with .NET

Although the options that have been discussed are available for the connection to the cube, they also can be used in conjunction with other .NET services and .NET Enterprise Servers to make data accessible from the Web and throughout an enterprise. This final section of the chapter describes the interaction between data warehousing and .NET services and then discusses the role of data warehousing in .NET Enterprise solutions.

.NET Services

Analysis Services is not as fully integrated with the .NET services as are many of the other products and services described in this book. It does, however, use some of the techniques and features that generally are associated with .NET. This section discusses the Web options available with Analysis Services to make OLAP cubes available to Web users. This is a beneficial solution for Internet– and intranet–based users.

One of the primary restrictions on the integration of Analysis Services and the Web is that Analysis Services requires the use of Windows authentication. The Web user has to be allowed to log on to a domain to access the data. Analysis Server has to be able to recognize the domain user. If the Web server is outside the firewall and the Analysis Server is inside the firewall, there is a need for a domain relationship between the two servers. This generally is accomplished with a trust relationship between the external domain and the internal domain. The developer should define the .NET Server relationship before deploying the data. The options for establishing the Windows authentication necessary for deploying Analysis Server data to the Web are outlined in the following list:

Note It is possible to connect to Analysis Server without having the user log on to a domain. This would have to be done through a single access to Analysis Server for all users. The developer could not take advantage of the role–based security in Analysis Server. This is a solution only if all the users need identical access to the data.

- **The Web server and the Analysis Server can both reside outside the firewall.** Both servers are on the same domain; therefore, Windows authentication can be used. This is the least common solution because most administrators feel that it puts their data at risk. Web users may be able to get direct access to Analysis Server. With this model the data warehouse still sits behind the firewall.
- **The Web server sits on a domain outside the firewall, and the Analysis Server sits on a domain inside the firewall.** The domain inside the firewall has a one–way trust

relationship with the domain outside the firewall. This relationship allows users from the outside domain to access resources on the inside domain, but not vice versa. To make this work it is necessary to allow the ports used for the trust relationship through the firewall. The port numbers used are dependent on the Active Directory implementation, and the developer should refer to that documentation to find the necessary ports to allow through the firewall.

- **The Web server and the Analysis Server are both on the same domain behind the firewall.** Web users have to interact with both machines. This can be accomplished by a couple of different means. First, one could allow port 80 (for a typical HTTP connection) or port 443 (for a Secure Sockets Layer connection) through the firewall. The second option, which is much less common but probably more secure, is to use server publishing from the firewall. With server publishing the Web users interact with the firewall, which then interacts with the data on the users' behalf. The Web users feel they have direct access to the servers when in fact they are only interfacing with the firewall. There is more information on server publishing in Chapter 14, "Internet Security and Acceleration Server."

> **Note** Active Directory domains in the same forest create two−way transitive trust relationships with parent and child domains. The one−way trust relationship described in this book is between two domains, one inside the firewall and one outside the firewall, that are in separate forests. They are not in a parent−child domain relationship. If they were, it would require a two−way transitive trust relationship, which could compromise security severely.

After you have negotiated the communication through the firewall, you need to choose how the Web users will interact with the data. You can allow them access to the cube data by using OWC, a custom interface, or a thin Web client. Each of these options has downsides, which should be compared before either solution is implemented. The following section compares and contrasts these options. First, this section describes the options available through ASP.NET and OWC. It then outlines the options for a custom interface. The section concludes by outlining the solution described as the thin Web client, which uses Active Server Pages (ASP) and an HTTP connection.

Office Web Components

OWC are a set of ActiveX controls that can be implemented within Web pages to simulate some of the features of the Office products. For instance, the OWC PivotTable component is the Web component that represents the PivotTable Service used to access cube data from Excel. This allows the Web user to access PivotTable without having Excel loaded. With this data access option, the Web user connects to the Web server to be authenticated and to access the Web page and components. The Web user then is redirected to Analysis Server and accesses data directly from the Analysis Server. This facilitates fast data retrieval and allows the Web connection to take advantage of the caching features supplied by the PivotTable Service.

The use of OWC has a couple of disadvantages:

- OWC redirects the user straight to Analysis Server. This requires both the Analysis Server and the Web server to be accessible to the user. Unless Server Publishing is being used, this requires the Analysis Server to be accessible from the Internet. This is accomplished by allowing the connection through the firewall or by placing the Analysis Server outside the firewall. Either choice could give the firewall administrator some security concerns.
- OWC requires Internet Explorer 4.01 or a later version. OWC will not work with another browser or an older version.
- OWC also requires the use of Microsoft Data Access Components (MDAC) 2.6. If the users do not have MDAC 2.6, the download is significant. If you still have NT 4.0 users, they have

to install SP6 to be able to install MDAC 2.6. The underlying software requirements may be large for the users.

Custom Interface

ADO MD can be used to query the cube data directly. The MD consists of the extensions to ADO that support multidimensional data sources and MDX as a query language. There is an OLE DB provider for Analysis Server, MSOLAP, that makes it possible to query the data and communicate the data to the front-end application. The OPENROWSET command can be used to expose the cube directly. There is more information about OPENROWSET in *SQL Server Books Online*.

The major disadvantage of this solution is the overhead required for its creation. You have to create the custom interface and may have difficulty emulating the features that are built into a PivotTable in Excel. Several vendors have created products that do this for you. All of them have features that enhance user reporting and analysis functionality with cubes. They also all come with a price tag, which the purchaser should evaluate.

Thin Web Client

Microsoft has made an Analysis Server thin Web client available. The thin Web client can be accessed from the SQL Server Resource Kit. The thin Web client uses ASP to connect to the Analysis Server and converts the multidimensional data to HTML. The HTML then is displayed to the user. The user is not redirected to the Analysis Server, and so MDAC 2.6 is not required with this solution. The thin Web client uses JavaScript and puts the queried data in a grid-like HTML table structure. The users have the ability to navigate through the cube, similar to the use of a PivotTable. The thin Web client also has some disadvantages, as all these solutions do:

- The thin Web client requires IE 5.0 or a later version.
- The thin Web client does not connect directly to Analysis Server. From a security standpoint this is a positive, but from a performance standpoint it is a negative. Your queries will appear much more sluggishly to users.
- The thin Web client does not use the PivotTable Service caching. This further decreases performance because more round trips must be made to the server.

In summary, each of these solutions has advantages and disadvantages. Those who like the built-in features and performance of OWC should consider evaluating server publishing. If a developer can make this OWC secure, it is a preferred solution. Regardless of the situation, a developer should analyze all these options to determine which one is best for the organization's needs.

.NET Enterprise Solutions

Data warehousing and OLAP databases play a large role in enterprise application integration (EAI). They rarely are used for business-to-business (B2B) and business-to-consumer (B2C) solutions. This book focuses on the role of data warehousing with EAI solutions.

Most of the EAI solutions that have been presented so far are integrating applications that reside on various server architectures. This EAI solution is at the database level rather than the application level. This is one of the more powerful options for integrating the data within an organization. In many cases the need for integrating data is centered on the need to analyze the data from multiple applications.

For example, an organization may have several departments that are responsible for their own inventory tracking. Each of these systems is managed by the department and may be stored in different databases (SQL Server, Oracle, Sybase, and so forth). At headquarters there probably is a need for reporting on the inventory tracking throughout the organization. This can be tedious if the data are maintained within each of the departments. Although one can get the reports from each department and try to summarize them, it would be better to summarize the data and then report on them. Data warehousing was created to facilitate this type of reporting.

DTS of SQL Server can be used to transfer the data to a central data warehouse and then build the necessary cubes for analysis and reporting. DTS can be used to transfer data to and from most database management systems and therefore provides a solid basis for satisfying data transformation needs.

One can view data warehousing as an EAI tool at the database level instead of the traditional application layer. This is a powerful way to summarize the data one needs to analyze for enterprisewide reporting requirements.

Because data warehousing is a tool at the database level and not the application level, the strategies for making the data available on the Web become slightly more complicated. It is necessary to interact with other .NET Servers and services to provide the full Web functionality without the risk of downtime or lost data.

In most cases organizations want access to their data in a secure manner. In addition, they have the following requirements for their data warehouse data:

- **The connection has to be available.** When Web users want to analyze the data, they should be able to connect to the server.
- **The data should be available.** Because the Web interaction separates the application from the data, the availability of the Web server for a connection does not guarantee that the data are available. This is complicated by the fact that Analysis Server is not a cluster–aware application.
- **The data retrieval should be fast.** With the data being available on the Web, most users will expect the application to be slower than a normal desktop application. The problem arises when the application is slower at some times than it is at others. This often occurs with Web–based applications because the number of application users increases. The solution should scale without a significant performance hit.

This leaves the task of extending the data warehouse to the Web while continuing to meet these requirements. The following example, as shown in Figure 17.5, provides a possible solution for this task.
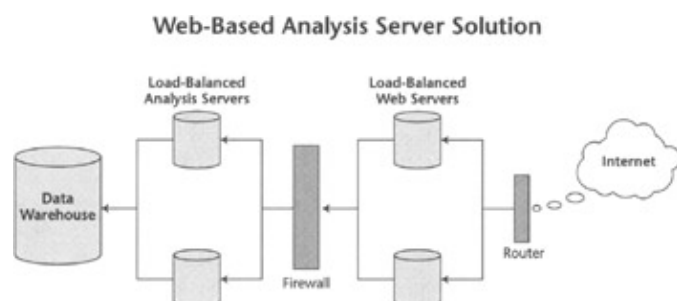
Figure 17.5: An example of Analysis Server cubes deployed to the Web.

In this case we are creating a solution to make the sales analysis data available to the Web in a secure, fast, and reliable fashion. At the Web server level we are using ASP and the thin Web client

solution for Analysis Services to allow the connection to the database. We are implementing two Web servers that are running Internet Information Server (IIS) and Application Center 2000. Through Application Center we implement load balancing. This helps ensure that the connection is available. It also helps the application scale. Application Center also is used to synchronize the ASP files between the two Web servers. For more information on load balancing and synchronization with Application Center 2000, refer to Chapter 9, "Application Center 2000."

The ASP being employed at the Web server level is used to connect to the Analysis Server that is behind the firewall. We need to make sure the data are always available for the connection. For this purpose, we also implement load balancing at this level. The cluster service generally is used for data availability, but one must keep in mind that Analysis Server is not cluster–aware. This makes it difficult, if not impossible, to cluster the Analysis Server. We are implementing two Analysis Servers that are load–balanced. For this to work effectively we must process identical cubes on both machines. This does duplicate the data on both boxes. The cost is drive space. It probably is worth it to help guarantee stability and performance.

The cubes that are used on the Analysis Servers are processed from the data warehouse. We have not implemented the cluster service or load balancing at the data warehouse level. This choice was made because the cubes are processed at night, and the Web users are not directly dependent on the data warehouse for their queries and reports. Furthermore, the data warehouse consists of data that have been transferred from transactional processing databases somewhere else in the organization. If we have problems with the data warehouse, we can always transfer the data from the transactional systems again. If we were using the cluster service, this would be done would be at the transactional level.

This example demonstrates the ability to move the Analysis Server cubes to an Internet medium. With this solution there are two concerns:

- We cannot use the write–back feature of Analysis Server. Write–back allows users to write data back to the cube. If one tries to implement this, the two Analysis Servers will be out of sync, and the analysis will not be consistent.
- We have to process the Analysis Server cubes identically. We should process them at about the same time and in the same manner. They will have to be stored in Multidimensional Online Analytical Processing (MOLAP) to ensure that the data are stored on the Analysis Server, not at the data warehouse. For more details on the storage options available with cubes, refer to *SQL Server Books Online*.

Are there other options? Of course. The key to this process is integrating the Web–based features of Analysis Server with the other .NET Enterprise Servers and services that are outlined throughout this book.

Review Questions

1. What are the core components of Analysis Server?
2. What is the purpose of data mining?
3. What is the difference between a data warehouse and OLAP?
4. Why should a developer use a star schema instead of a snowflake schema when designing a data warehouse?
5. Why is the grain of the fact table important?
6. What is a cube?
7. What options are available for making cube data available on the Web?

Chapter 18: Clustering .NET Servers

Highlights

Clustering is a technology that involves linking individual servers physically and programmatically to coordinate communication between them. The clustered servers can work together to perform a single task or operation. If any server stops functioning properly, an event referred to as a failover, its workload automatically shifts to another server so that continuous service can be provided. In addition to the failover process, some clustering technologies provide network load balancing (NLB), which enables the workload to be shared across the linked servers.

Clustering services combine with the large memory support and multiprocessing functionality of .NET Advanced Server and Datacenter Server to enable organizations to ensure the availability of critical applications while being able to scale those applications up and out to meet increased demand.

This chapter describes the cluster services available in the .NET Server platform. The first part provides an overview of the clustering services and includes a description of NLB, which is categorized as a cluster service. This section identifies the reasons for using a cluster and the new features included in the .NET cluster environment. The chapter then outlines the cluster architecture. This section defines the core components of a cluster and the tools used to configure and monitor a cluster. The chapter concludes with a section about implementing a cluster solution and its impact on the .NET Enterprise suite of products. This final section outlines the basic administrative tasks of configuring the cluster and then describes the process of implementing an SQL Server and an Exchange Server in a cluster.

Overview of Clustering

A cluster is a group of independent computers that are linked together to run a single application or operation. While the servers are independent, the primary goal of the cluster is to make the servers in the cluster appear as a single entity to the end user. The computers are connected physically by cables and programmatically by cluster software. Clustering and NLB typically are not implemented by the developer of an application. The application can be installed in a cluster or on servers that are load–balanced to help with performance and availability. These connections allow computers to use failover and load balancing, something that is not possible with a standalone computer. .NET clustering technology provides the following benefits:

- **High availability.** A cluster avoids a single point of failure. Applications can be distributed over more than one computer, achieving a degree of parallelism and failure recovery and providing more availability.
- **Scalability.** A cluster's computing power can be increased by adding more processors or computers.
- **Manageability.** A cluster appears as a single–system image to end users, applications, and the network while providing a single point of control to administrators. The single point of control can be remote.

Types of Clustering

In the .NET Advanced Server and Datacenter Server operating systems, Microsoft has introduced two clustering technologies that can be used independently or in combination, providing a complete

set of clustered solutions that can be selected on the basis of the requirements of a given application or service. The .NET clustering technologies include the two following services:

> **Note** Application Center Server also supports clustering. This is not the same clustering technology that is the focus of this chapter. In Application Center the purpose of the cluster is to group computers together. This can be done to facilitate configuration, build in fault tolerance, and load–balance Web servers. Although these purposes are similar to the advantages of the clustering services of .NET Server, the services are different. More information on the clustering of Application Center Server appears in Chapter 9, "Application Center 2000."

- **Cluster service.** This service is intended primarily to provide failover support for applications that are running mission–critical applications. These applications could include databases, messaging systems, and file or print services. Cluster service supports two–node failover clusters in .NET Advanced Server and four–node clusters in Datacenter Server. Two–node and four–node clusters are defined in more detail in the section titled *Cluster Architecture*. The purpose of the cluster service is to provide fault tolerance at the data level. Therefore, the cluster service typically is implemented with products that are very database–dependent, such as SQL Server and Exchange Server. Without their databases these products are not very useful.
- **Network load balancing.** This service load–balances incoming Internet Protocol (IP) traffic across clusters with up to 32 nodes. NLB enhances both the availability and the scalability of Internet server–based programs such as Web servers, streaming media servers, and terminal services. By acting as the load–balancing infrastructure and providing control information to management applications built on top of Windows Management Instrumentation (WMI), NLB can be integrate seamlessly into existing Web server farm infrastructures. NLB is the key to the implementation of Application Center Server. There is more information on configuring NLB for Application Center Server in Chapter 9, "Application Center 2000." In contrast to the cluster service, NLB provides fault tolerance at the connection or network availability level. It does not provide fault tolerance for data. Therefore, NLB typically is implemented on products that are more concerned with user connectivity than with data storage and retrieval. These products include Application Center 2000, Internet Information Server (IIS), and Internet Security and Acceleration (ISA) Server.

Both Windows clustering technologies can be used in conjunction to create highly scalable and available e–commerce sites. When NLB is deployed across a front–end Web server farm and back–end line–of–business applications such as databases are clustered with luster service, one gains all the benefits of scalability with a low risk of failure at the database level.

For example, say you need to deploy an analysis package that is Web–based. The purpose of the package is to allow users to connect to a data warehouse and analyze customer history information. The application has been written in ASP.NET. The Active Server Pages (ASP) reside on a Web server, and the data are stored in SQL Server. The ASP are written to access the data from the SQL server by using ADO.NET. In this example, if you install one Web server and one SQL Server, you have two single points of failure. You could use NLB to load–balance multiple Web servers to help guarantee availability. The ASP have to be installed on all the Web servers. You then could deploy SQL Server in a cluster with two physical servers maintaining fault tolerance at the data level. In this example you have used both of the services to get the best of both worlds. Table 18.1 identifies some of the .NET Enterprise solutions and the service that is appropriate in each case.

Table 18.1: Comparison of Clustering Technologies

ENTERPRISE SOLUTION	CLUSTER SERVICE	NETWORK LOAD BALANCING	BENEFITS
Terminal services	No	Yes	Improved scalability and availability
Database servers	Yes	No	Minimize downtime and ensure availability
Messaging servers	Yes	No	Minimize downtime and ensure availability
Web server farm	No	Yes	Expand capacity
E–Commerce sites	Yes	Yes	Expand and scale without risk of failure

New Cluster Features in .NET Server

Clustering services have been available since NT 4.0 Enterprise Edition. They were not used commonly at that stage, partly because one also had to have the Enterprise Edition of NT Server installed, which not many users did. The service was enhanced and implemented much more on the Windows 2000 platform. The .NET platform provides several enhancements to its predecessor related to clustering. Here are the new features of .NET Server clustering:

- **Active Directory support for virtual servers.** This feature provides a computer object in Active Directory for cluster servers; this means that cluster–aware and Active Directory–aware applications have an object in Active Directory. Administrators can use this feature to find virtual server computer objects in the Active Directory. This is somewhat limited. Viewing of cluster topology information or the association between virtual servers and clusters is not supported at this time.
- **Network load balancing for virtual clusters.** .NET Server provides the ability for each hosted application, Web site, or virtual IP address in an NLB cluster to specify its own set of port rules. By enabling this capability, virtual clusters allow traffic to different virtual IP addresses in the same NLB cluster to be handled differently, based on the destination virtual IP address. As a result, virtual clusters can be managed independently.
- **Eight–node clusters.** In .NET Server maximum cluster service size has been increased from a four–node to an eight–node cluster in Windows .NET Datacenter Server. This provides increased flexibility for adding and removing hardware in a geographically dispersed cluster environment as well as improved scaling options for applications. As a result, information technology (IT) administrators can work with their application developers to deploy a large–scale application on an eight–node cluster that is geographically dispersed in two locations. The application also will benefit from improved failover management resulting from the eight–node configuration.
- **Bidirectional affinity for ISA Server clusters.** To enable Microsoft ISA Server clustering using NLB, NLB has to operate in a special mode that ties the NLB instances running on the inside and outside network interfaces of ISA Server and allows the NLB on the internal interface of ISA Server to hash on a connection's destination IP address instead of the usual source IP address. This feature is useful for Web and server publishing using NLB and ISA Server.
- **Quorum of nodes.** This feature provides a quorum device that is accessible to all the nodes in a cluster. This disk is used to arbitrate when there are communication failures and as a central location in which to store highly available configuration data. In practice, IT administrators with a geographically dispersed cluster can use this feature to support multiple sites without the need for shared storage technology to span the sites.

- **Changing passwords without reboot.** Windows .NET Server provides a tool that allows the cluster account to be updated while the cluster service remains online. IT administrators can use this feature to change passwords on a node without rebooting the server; this results in improved availability and uptime.
- **Individual group in cluster availability metrics.** This feature provides a means to assess the availability of an individual group in a cluster. The cluster service will log group moves and online or offline events for success and failure in the system event log. In addition, internode clock skew events are written to the system event log. By analyzing the event log streams from all the cluster nodes that look at the times between offline and online events (taking into account cross–node clock skews), administrators can calculate the amount of time a group is online versus offline. Log cluster availability events are added to the event log with adequate timing information to allow calculation of the availability of a cluster resource group.

Cluster Architecture

The cluster architecture is not difficult, but it is different from other Microsoft technologies and sometimes is frustrating because the concepts are so new to users who are trying to implement the cluster. The following sections introduce the architecture. After the identification of a few key terms that should be defined, the configuration options of the cluster are addressed. Finally, the core components and tools available for managing and configuring the cluster are considered.

Key Terms

This section defines a few key terms that describe the cluster architecture:

- **Cluster service.** Cluster service is the Windows 2000 name for the Microsoft technology first made available as Microsoft Cluster Server (MSCS) in Windows NT Server 4.0, Enterprise Edition. When one is referring to the servers that constitute a cluster, individual computers are called nodes. Cluster service refers to the collection of components on each node that perform cluster–specific activity, and resource refers to the hardware and software components within the cluster that are managed by cluster service. The instrumentation mechanism provided by cluster service for managing resources is the resource dynamically linked libraries (DLLs), which define resource abstractions, communication interfaces, and management operations.
- **Node.** A node is an individual server in a cluster.
- **Resource group.** A resource group is a collection of resources managed by cluster service as a single, logical unit. Application resources and cluster entities can be managed by grouping logically related resources into a resource group. When a cluster service operation is performed on a resource group, the operation affects all the individual resources in the group. Typically, a resource group is created to contain all the elements needed by a specific application server and client for successful use of the application.
- **Quorum resource.** A special common resource is the quorum resource, a physical disk in the common cluster disk array that plays a critical role in cluster operations. It must be present for node operations such as forming and joining a cluster to occur.

Configuration Options

The cluster service uses the standard .NET drivers for local storage devices and media connections. The cluster service also supports several different types of connection media for the

external drive array that must be accessible to all the servers in the cluster. External storage devices that are common to the cluster require SCSI devices and support standard PCI–based SCSI connections as well as SCSI over Fibre channel and an SCSI bus with multiple initiators. Fibre connections are SCSI devices that are hosted on a Fibre channel bus instead of an SCSI bus. Conceptually, Fibre channel technology encapsulates SCSI commands within the Fibre channel and makes it possible to use the SCSI commands that cluster service supports.

Figure 18.1 illustrates the two–node cluster configuration. The cluster can be configured in a two–node, four–node, or eight–node configuration. The two–node cluster is available in .NET Advanced Server; .NET Datacenter Server is required for a four–node or eight–node cluster. In a two–node cluster there is the choice of an active/active or an active/passive configuration. The differences between these two options are as follows:

Two-Node Cluster Configuration
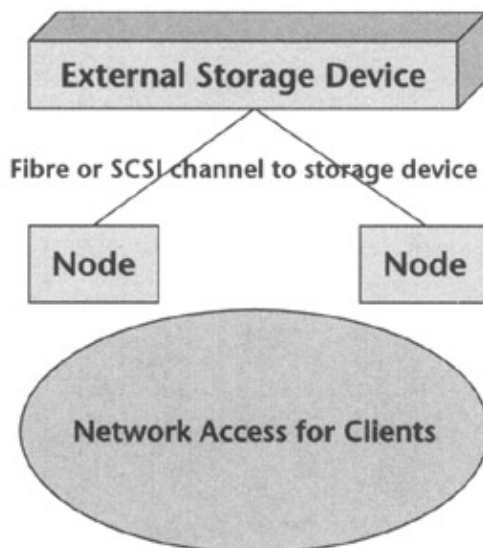
Figure 18.1: In this two–node cluster the two servers are configured to use the external storage device for data storage.

- **Active/active cluster.** In an active/active cluster both of the nodes (servers) can process client requests actively. When one of the servers fails and the failover process occurs, the server that takes the additional processes will perform the actions of both servers. This can result in slower performance after a failover.
- **Active/passive cluster.** In an active/passive cluster one of the nodes is servicing client requests actively. The other node is in a standby mode. When the server that is primary fails over, the secondary server picks up the processing. In this configuration there is always one server that is not processing client requests.

Windows 2000 Datacenter Server supports four–node and eight–node clusters and requires a fibre channel for the connection to the external storage device. In a four–node or eight–node configuration there is more flexibility in the choice of the use of each of the servers. Some of the following items can be used as a strategy for implementing four–node or eight–node clusters:

- All the servers can be active. In this configuration all the servers are actively servicing client requests. When a server fails over, the processing moves to another server or set of servers in the cluster.
- All the servers except one are active, and a single node is reserved as passive. In this

configuration there is one node that is in standby mode. If one of the other nodes fails, its processing falls to the standby node.
- Half the servers can be active, and half can be passive. This is the most expensive solution, but it provides maximum fault tolerance.

When one is configuring a cluster, two key features help with the management of the resources that are assigned to the cluster. Virtual servers are used to expose an application or set of data to users as though it were a single physical server. Resource groups are logical groupings of cluster resources. The following sections describe these two key features.

Virtual Servers

One of the benefits of cluster service is that applications and services that are running on a server cluster can be exposed to users and workstations as virtual servers. To users and clients, connecting to an application or service that is running as a clustered virtual server appears to be the same process as connecting to a single physical server. In fact, any node in the cluster can host the connection to a virtual server. The user or client application will not know which node actually is hosting the virtual server.

In there is an application or server failure, cluster service moves the entire virtual server resource group to another node in the cluster. When such a failure occurs, the client will detect a failure in its session with the application and attempt to reconnect in exactly the same manner in which the original connection was attempted. It will be able to do this successfully because cluster service simply maps the published IP address of the virtual server to a surviving node in the cluster during recovery operations. The client session can reestablish the connection to the application without needing to know that the application now is hosted physically on a different node in the cluster.

Note that while this provides high availability of the application or service, session state information related to the failed client session is lost unless the application is designed or configured to store client session data on disk for retrieval during application recovery. Cluster service enables high availability but does not provide application fault tolerance unless the application itself supports fault–tolerant transaction behavior. Microsoft Dynamic Host Configuration Protocol (DHCP) service is an example of an application that stores client data and can recover the data from failed client sessions. DHCP client IP address reservations are saved in the DHCP database. If the DHCP server resource fails, the DHCP database can be moved to an available node in the cluster and restarted with restored client data from the DHCP database.

Resource Groups

Resource groups are logical collections of cluster resources. Typically, a resource group includes logically related resources such as applications, their associated peripherals, and data. However, resource groups can contain cluster entities that are related only by administrative needs, such as an administrative collection of virtual server names and IP addresses. A resource group can be owned by only one node at a time, and individual resources within a group must exist on the node that currently owns the group. At any given instance, different servers in the cluster cannot own different resources in the same resource group.

Each resource group has an associated clusterwide policy that specifies which server the group prefers to run on and which server the group should move to in case of a failure. Each group also has a network service name and address to enable network clients to bind to the services provided by the resource group. If there is a failure, resource groups can be failed over or moved as atomic units from the failed node to an available node in the cluster.

Each resource in a group may depend on other resources in the cluster. Dependencies are relationships between resources that indicate which resources have to be started and available before another resource can be started. For example, a database application may depend on the availability of a disk, IP address, and network name to be able to start and provide services to other applications and clients.

Resource dependencies are identified by using cluster service resource group properties and enable cluster service to control the order in which resources are brought online and offline. The scope of any identified dependency is limited to resources in the same resource group. Cluster–managed dependencies cannot extend beyond the resource group because resource groups can be brought online and offline and moved independently.

Cluster Service Components

Cluster service runs on the .NET Server operating system by using network drivers, device drivers, and resource instrumentation processes designed specifically for cluster service and its component processes. The following components constitute the core of the cluster architecture:

- **Checkpoint Manager.** Saves application registry keys in a cluster directory stored on the quorum resource.
- **Communications Manager.** Manages communications between cluster nodes.
- **Configuration Database Manager.** Maintains cluster configuration information.
- **Event Processor.** Receives event messages from cluster resources, such as status changes and requests from applications to open, close, and enumerate cluster objects.
- **Event Log Manager.** Replicates event log entries from one node to all the other nodes in the cluster.
- **Failover Manager.** Performs resource management and initiates appropriate actions, such as startup, restart, and failover.
- **Global Update Manager.** Provides a global update service used by cluster components.
- **Log Manager.** Writes changes to recovery logs stored on the quorum resource.
- **Membership Manager.** Manages cluster membership and monitors the health of the other nodes in the cluster.
- **Node Manager.** Assigns resource group ownership to nodes on the basis of group preference lists and node availability.
- **Object Manager.** Manages all the cluster service objects.
- **Resource monitors.** Monitor the health of each cluster resource by using callbacks to resource DLLs. Resource monitors run in a separate process and communicate with the cluster service through remote procedure calls (RPCs) to protect the cluster service from individual failures in cluster resources.

Implementing the Cluster Solution

Implementing the cluster solution involves configuring the hardware needed to support the solution. As was described earlier in this chapter in the section titled _Cluster Architecture_, it is necessary to have the external storage device and the nodes that will form the cluster. A developer should have the hardware installed and configured before addressing the software configuration.

The following sections describe the software implementation of the cluster solution. The first section describes the process of creating the cluster. The next section describes the other basic administrative tasks related to configuring the cluster and explains the forming, joining, and leaving of a cluster. The section after that discusses installing applications on a cluster. The considerations

for installing SQL Server and Exchange Server in a cluster environment are described.

Creating a Cluster

Cluster service includes a cluster installation utility to install the cluster software on a server and create a new cluster. To create a new cluster, the utility is run on the computer that has been selected to be the first member of the cluster.

This step defines the new cluster by establishing a cluster name and creating the cluster database and the initial cluster membership list. The cluster installation utility is included with Windows 2000. The cluster software is built into the Advanced Server and Datacenter Server operating systems. The name of the cluster is defined as the cluster service is installed. The cluster service is installed as an additional operating system component. This can be performed during the installation of Windows 2000 or at a later time by using the Add/ Remove Programs icon in the control panel. After the name of the cluster is defined, the installation utility copies the needed files from the Windows 2000 CD–ROM. More information on the addition of the cluster service is available in the Windows 2000 help files.

The next step in creating a cluster is adding the common data storage devices that will be available to all the members of the cluster. This establishes the new cluster with a single node, with its own local data storage devices, and with the cluster common resources, generally disk or data storage and connection media resources. Adding the common data storage is a process that is specific to the type of storage that is being added. Refer to the product documentation for the specifics of adding the storage device.

The final step in creating a cluster is running the installation utility on each additional computer that will be a member of the cluster. As each new node is added to the cluster, it automatically receives a copy of the existing cluster database from the original member of the cluster. When a node joins or forms a cluster, cluster service updates the node's private copy of the configuration database. In adding a second node to the cluster it is necessary to specify the name of the cluster that is being joined, as shown in Figure 18.2. If you are physically connected to the cluster, the installation process loads the necessary files and starts the cluster service.
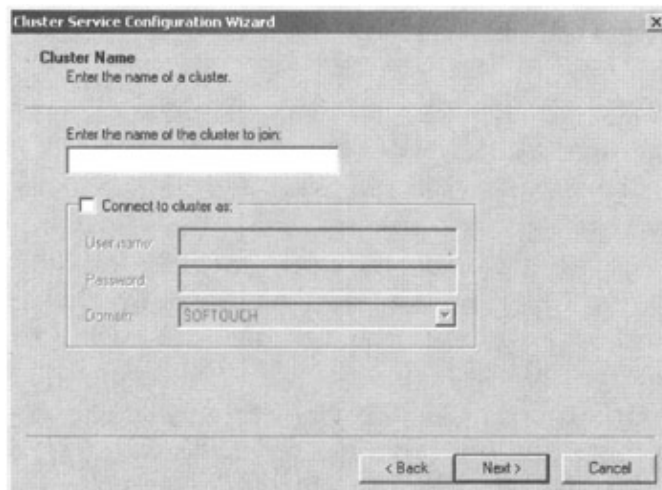
Figure 18.2: Additional nodes in a cluster are added by installing the cluster service on the additional machines and specifying the name of the cluster that is being joined.

Administration of the Cluster

After the cluster is set up, some basic administration functions have to be performed to allow servers to act as nodes in the cluster. The following sections outline the core administrative tasks related to administering the cluster. The details of forming, joining, and leaving the cluster are described in more detail.

Forming a Cluster

A server can create a new cluster if it is running cluster service and cannot locate other nodes that already are using the desired cluster name. This process helps ensure that duplicate clusters are not created on the network. To form the cluster, a node must be able to acquire exclusive ownership of the quorum resource. That resource maintains data integrity and cluster unity and plays a critical role in cluster operations. It must be present for node operations such as forming and joining a cluster to occur. The quorum resource is a physical disk in the common cluster disk array and has the following attributes:

- Supports low–level commands for persistent ownership arbitration, enabling a single node to gain and defend physical control of the quorum resource. For example, the SCSI disk Reserve and Release commands enable persistent arbitration.
- Can be accessed by any node in the cluster.
- Can be formatted by using NTFS.

The quorum resource performs the role of a tiebreaker when a cluster is formed or when network connections between nodes fail. When a cluster initially is formed, the first node in the cluster contains the cluster configuration database. As each additional node joins the cluster, it receives and maintains its own local copy of the cluster configuration database. The quorum resource on the common cluster device stores the most current version of the configuration database in the form of recovery logs that contain node–independent cluster configuration and state data. During cluster operations the cluster service uses the quorum recovery logs to do the following:

- Guarantees that only one set of active, communicating nodes is allowed to form a cluster.
- Enables a node to form a cluster only if it can gain control of the quorum resource.
- Allows a node to join or remain in an existing cluster only if it can communicate with the node that controls the quorum resource.

From the point of view of other nodes in the cluster and the cluster service management interfaces, when a cluster is formed, each node in the cluster may be in one of three distinct states. These states are recorded by the event processor and replicated by the event log manager to other clusters in the node. The cluster service states are as follows:

- **Offline.** The node is not a fully active member of the cluster. The node and its cluster service may or may not be running.
- **Online.** The node is a fully active member of the cluster. It honors cluster database updates, contributes votes to the quorum algorithm, maintains heartbeats, and can own and run resource groups. All the servers in the cluster determine the total number of nodes that are currently online by using the quorum algorithm. The heartbeat is an identification packet that is sent from a node in the cluster to notify the other nodes that it is still available.
- **Paused.** The node is a fully active member of the cluster. It honors cluster database updates, contributes votes to the quorum algorithm, and maintains heartbeats but cannot accept resource groups. It can support only those resources groups for which it currently has ownership. The paused state is provided to allow certain types of maintenance to be

performed. Online and paused are treated as equivalent states by the majority of the cluster service components.

Joining a Cluster

To join an existing cluster, a server must be running the cluster service and must locate another node in the cluster. After finding another cluster node, the joining server must be authenticated for membership in the cluster and must receive a replicated copy of the cluster configuration database.

The process of joining an existing cluster begins when the Windows 2000 or Windows NT Service Control Manager starts cluster service on the node. During the start–up process cluster service configures and mounts the node's local data devices. It does not attempt to bring the common cluster data devices online as nodes because the existing cluster may be using those devices.

To locate other nodes, a discovery process is started. When the node discovers any member of the cluster, it performs an authentication sequence. The first cluster member authenticates the newcomer and returns a status of success if the new server is authenticated successfully. If authentication is unsuccessful—a joining node is not recognized as a cluster member or has an invalid account password—the request to join the cluster is refused.

After successful authentication, the first node online in the cluster checks the copy of the configuration database on the joining node. If that database is out of date, the cluster node that is authenticating the joining server sends to it an updated copy of the database. After receiving the replicated database, the node joining the cluster can use it to find shared resources and bring them online as needed.

Leaving a Cluster

A node can leave a cluster when it shuts down or when the cluster service is stopped. However, a node also can be forced to leave (be evicted) when the node fails to perform cluster operations, such as failure to commit an update to the cluster configuration database.

When a node leaves a cluster as part of a planned shutdown, it sends a ClusterExit message to all the other members in the cluster, notifying them that it is leaving. The node does not wait for any responses and immediately proceeds to shut down resources and close all cluster connections. Because the remaining nodes received the exit message, they do not perform the same regroup process to reestablish cluster membership that occurs when a node unexpectedly fails or network communications stop.

When a node is evicted, for example, by a manual operation from Cluster Administrator, the node status is changed to evicted. An eviction is a forceful removal from the cluster.

.NET Enterprise Servers in a Cluster Environment

After the cluster hardware is in place and the servers have been configured as nodes in the cluster, the developer can install software that can take advantage of the cluster services. An application must be cluster–aware to take advantage of the cluster services.

At the time of the writing of this book not many client/server applications are cluster–aware. One can expect the list of applications that are cluster–aware to expand as the technology is implemented more widely.

.NET Enterprise Servers can be tightly integrated into the cluster service. Microsoft SQL Server and Exchange Server are both cluster–aware applications. Both are applications that are very database–dependent, and this makes them great candidates for a cluster. Keep in mind that NLB is used to provide fault tolerance for the connections to the servers. This is common for IIS, Commerce Server, and Application Center 2000. The cluster service is used to provide fault tolerance at the database level. Because of the nature of their services, SQL Server and Exchange Server generally are considered database–dependent and are good candidates for clustering. The following sections describe the process of installing and configuring SQL Server and Exchange server on a cluster.

SQL Server Cluster Configuration

After MSCS has been installed and configured, the next step is to configure SQL Server for clustering. SQL Server 2000 is cluster–aware and is designed to make use of clustering. This section focuses primarily on the different design issues in running SQL Server with the cluster service.

Planning the Deployment

The first step in planning an SQL Server cluster is to determine the type of hardware to use and the mode of operation in which the cluster will run. It is necessary to choose whether the implementation of cluster service for SQL Server will be active/passive or active/active:

- Active/passive cluster configurations should consist of computers with identical hardware. Each of these machines should be capable of handling the workload of the databases. Because active/passive mode does not use the secondary system during normal operations, the performance of the SQL Server instance should remain constant. Users will not experience any performance change if the primary system fails over to an identical secondary system. With an active/passive cluster one can choose to invest in the additional hardware to guarantee the performance of the application. The only performance hit will be the time it takes for the failover process. In most cases this should not exceed 90 seconds. With an active/passive cluster you have configured a standby server that will be used when the primary server becomes unavailable.
- Active/active cluster configurations should consist of two systems, each of which is running a specific workload. The configuration and performance will be optimal if both systems have identical hardware. If a failure occurs, the surviving system will take over the workload of the failed system. This will cause the load of both servers to run on one server. If either of these servers had a high utilization rate before the failover, this process most likely will result in decreased performance. With active/active clustering one typically is choosing to have a fault–tolerant solution at the expense of performance. When the system performs a failover, the performance may be slower, but all the data will be available to the users. The developer generally should document the configuration of the cluster and inform users and other support staff that a failover will result in a performance decrease.

The next step to perform in configuring SQL Server for a cluster is to check and possibly change several SQL Server settings. The next sections examine those settings.

Setting the Recovery Time

The recovery time interval is used to configure the amount of time the developer wants SQL Server to take to recover a database after a system failure. Recovery is the process of reading the log file from the last checkpoint and applying all completed transactions to the database and erasing all uncompleted transactions.

In tuning SQL Server, the developer may have set the configuration parameter recovery interval to something other than the default value of 0. Changing this setting will increase the time between checkpoints and improve the overall performance of the server but also will increase recovery time. (The system must recover after it has failed over.) While this is appropriate in some cases, it is not appropriate in working with a clustered server. In a clustered system, the default value of 0, which specifies automatic configuration, should be used. This may result in decreased performance because of the ongoing checkpoints, but the purpose of the cluster service is fault tolerance. The default setting of 0 for recovery time results in a checkpoint about every minute. This timing speeds the failover process and makes the data available to the users faster.

Configuring Min Server Memory

To create an active/passive cluster configuration it may be necessary to change one setting in SQL Server. If the secondary server is identical to the primary server, no change is necessary. If the secondary server has fewer resources than the primary server has, the SQL Server configuration parameter *min server memory* should be set to 0. This setting instructs SQL Server to allocate memory on the basis of available system resources and allows for the dynamic allocation of memory. It is necessary for the system to control the memory when the two servers are not identical from a hardware standpoint.

In an active/active cluster configuration it is necessary to set the SQL Server configuration parameter *min server memory* to 0. This allows the server to reconfigure the memory for each instance after the failover. If this is not configured, the server instance that has failed over to the other server will not be allocated any memory or the system will force everything the new instance does to the paging file. This will affect performance negatively.

Installing SQL Server for Clustering

When installing SQL Server for clustering, the developer should take the following steps in a manner similar to the normal installation of SQL Server 2000. The developer will come to the screen to identify whether the installation is for a local or a remote server and will have the option to select a virtual server, as shown in Figure 18.3.
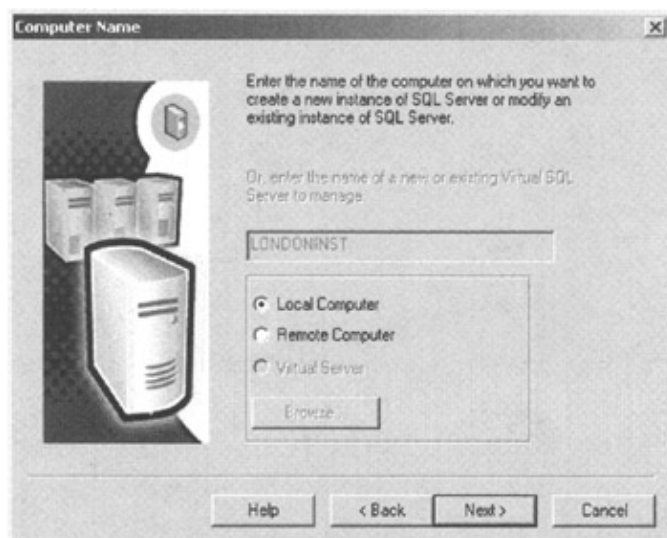
Figure 18.3: In installing SQL Server for clustering, it is necessary to install a virtual instance. (In this case the option is grayed out because cluster service was not configured first.)

The only other installation issue is the file paths. You will be prompted for the location of the program files and the system database file path, as shown in Figure 18.4. You want to make sure

that both of the paths are pointed to a drive located on the shared network storage device. This will ensure that the files are not kept locally and that the failover process can be successful.
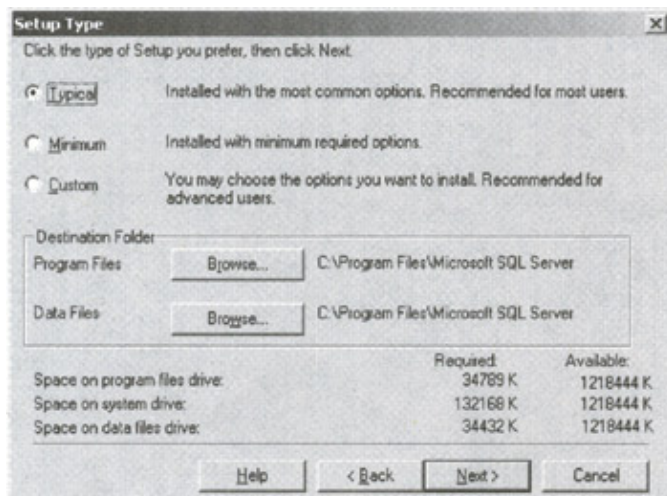
Figure 18.4: In installing a virtual instance of SQL Server, store the files on the shared storage device.

Exchange Server Cluster Configuration

Many Exchange 2000 Server components support active/active or at least active/passive clustering. However, several components, such as the Network News Transport Protocol (NNTP) service, the Key Management Server (KMS), and connectors to other mail systems, are not supported. Therefore, the developer should integrate an Exchange 2000 cluster into an environment with non−clustered servers functioning as bridgehead servers to communicate with other systems. This is an acceptable approach because permanent availability is seldom an issue with bridgehead servers that do not host mailboxes or public servers.

Bridgehead server is a term that describes a server that coordinates the communication to another physical location (routing group) or another type of mail system. A bridgehead server allows the flow of the messaging to be funneled through a single server. This decreases the configuration and overhead of the servers that are not functioning as bridgehead servers. In many cases bridgehead servers act only as a gateway to another location or mail system and do not house any mailboxes or public folders. Similarly, it is not advantageous to install front−end servers in clustered systems that use the Windows 2000 cluster service. Because front−end servers relay Internet−based clients only to back−end servers, one should use Windows 2000 NLB or another network load−balancing solution instead.

The installation of Exchange 2000 Server in a cluster is a four−step process:

1. Install Windows 2000 Advanced Server or Datacenter Server on all the nodes in the cluster.
2. Install the cluster service on the nodes and configure the cluster environment.
3. Install Exchange 2000 Server on all the nodes with exactly the same parameters.
4. Configure and start virtual servers in Cluster Administrator.

There is more information about the installation and configuration of Exchange Server 2000 in a cluster in the Exchange installation readme.txt file. This documentation is available on the Exchange Server 2000 CD−ROM.

Note A developer cannot install Exchange 2000 Server on a nonclustered server and integrate that installation into a cluster afterward. Furthermore, it is a good idea to test the installation in a

cluster before deploying Exchange 2000 Server in the production environment. Exchange 2000 Server may not function properly with your hardware. You need to ensure that a dedicated physical−disk resource is available for Exchange 2000 Server.

To install Exchange 2000 in a cluster, the installation program and the administrator need to set additional parameters and additional configuration options. The following sections describe the changes that must be made for Exchange Server to be installed in a cluster. The first section describes the installation options that have to be supplied to install Exchange in a cluster. The next section outlines the dependencies of the Exchange installation that are specific to the cluster. Then the reader is introduced to the option of installation by using an initialization file. Finally, a section describes the configuration of the virtual servers. Virtual servers provide access to the IPs in Exchange Server 2000.

Installing Exchange 2000 In a Cluster

It is easy to install Exchange 2000 Server in a clustered system. When the setup program detects the cluster service, it displays the Microsoft Exchange 2000 Installation Wizard, indicating that the cluster−aware version of Exchange 2000 Server will be installed.

If your installation does not pick up the cluster−aware version, stop the installation and verify the cluster server configuration. Setup then copies and configures the Exchange 2000 components and resource DLLs and sets the Exchange 2000 service to start manually. This prevents the services from starting automatically when one is rebooting the server, which is required at the end of the installation.

> **Note**   Do not start or stop clustered Exchange 2000 services in the services management tool. The Exchange Server services should be taken offline and online from the Cluster Administrator.

Clustered Installation Dependencies

During the installation of Exchange 2000 Server in a cluster, resource DLLs and other components are added to the configuration of each node. It is important to install only one node at a time, using the same account that was used to install the cluster service. This account has to have the appropriate permissions to install Exchange in a Forest. For more information on installing Exchange Server and the permissions required to perform the installation, refer to the Exchanger Server help documentation.

It is important to specify the same drive letters and directory on all the nodes in the cluster. During installation it is necessary to place the binary files on the local system drive of each node. The binary files are not shared between the nodes. Do not forget to make sure that drive M is not in use on any node because when the virtual servers are configured later, the Microsoft Web storage system will use the M drive by default. If this drive is unavailable, the Web storage system uses the next drive letter automatically; however, it is important that all nodes use the same drive. More information on the Web storage system appears in Chapter 7, "Exchange Server."

Setup Initialization File

Because the same Exchange 2000 components must be installed on all the nodes in the cluster, it is a good idea to create and use a SETUP.INI file. This allows you to run Setup unattended with exactly the same options on all nodes. Keep in mind, however, that the Setup program should not be started on a second node before the first installation is completed and the node is rebooted. At a minimum, install Microsoft Exchange Messaging and Collaboration and Microsoft Exchange System

Management tools on all nodes.

As soon as you have installed Exchange 2000 Server on all cluster nodes, you are ready to configure resource groups. Each virtual server (equivalent to a resource group) requires an IP address and a network name. The users will specify the network name in the settings of the Exchange transport service to connect to their mailboxes. Each virtual Exchange 2000 server requires one or more shared disk resources where the information store databases must be placed. You cannot assign a single physical disk to more than one virtual server. All Exchange 2000 components depend on the system attendant (SA), and so it also is necessary to assign the virtual server an Exchange SA resource. The remaining Exchange 2000 components are added to the virtual server automatically.

Depending on the load-balancing and failover strategy, it may be desirable to configure multiple virtual Exchange 2000 servers in a cluster. Each virtual server will appear in the network as a separate server. For each server, one can specify preferred owner information to distribute the resources equally across the cluster nodes.

Virtual Protocol Servers

As soon as you have configured a virtual Exchange 2000 server, you can use the Exchange System Manager to configure additional virtual servers for Internet access protocols. However, do not use the System Manager to bring a virtual server online. Cluster Administrator is the right utility to complete the configuration of virtual protocol servers. Additional virtual servers allow you to create support for multiple domains and are beneficial in creating some security strategies in Exchange Server 2000. There is more information on the configuration of virtual servers in the Exchange Server help documentation.

Review Questions

1. What are the advantages of implementing the cluster services of .NET Server?
2. What are the differences between the cluster service and network load balancing?
3. What is the role of the virtual server?
4. What is a resource group?
5. What is the role of the external drive array in a cluster?
6. How many nodes are supported for a cluster configuration?
7. Why is it necessary to install Exchange Server in all the nodes in a cluster?

Part IV: Appendixes

Appendix List

Appendix A: Visual Basic .NET Reference Guide

Overview

Visual Basic .NET, the current generation of the Visual Basic language, provides a fast and easy way to create .NET applications, including XML Web services and Web applications. Visual Basic .NET has many new and improved features that make it a powerful object–oriented programming language, including inheritance, interfaces, and overloading. Other new language features include free threading and structured exception handling. Visual Basic .NET also fully integrates the .NET Framework and the common language runtime, which provide language interoperability, garbage collection, enhanced security, and improved versioning support.

The following sections introduce the essential components of Visual Basic .NET. After creating the interface for an application by using forms and controls, it is necessary to write the code that defines the application's behavior. As with any modern programming language, Visual Basic supports a number of common programming constructs and language elements.

For readers who have programmed in other languages, much of the material covered here will seem familiar. While most of the constructs are similar to those in other languages, the event–driven nature of Visual Basic introduces some subtle differences.

For those who are new to programming, the material in this section will serve as an introduction to the basic concepts for writing code. Once the reader understands the basics, he or she can create powerful applications by using Visual Basic.

Variables

Visual Basic, like most programming languages, uses variables to store data temporarily. A variable has the following properties:

- **Name.** Identifier used to access the variable in code.
- **Address.** Memory address where the variable value is stored.
- **Type.** Data type of the variable.
- **Value.** Value stored in the variable's memory address.
- **Scope.** Level of visibility of variable through the program.
- **Lifetime.** How long a variable exists.

A developer declares a variable to specify its name and data type. The declaration statement for variables is the Dim statement. Its location and contents determine the variable's characteristics.

Declaration Levels

A local variable is a variable that is declared within a procedure. A module variable is declared at the module level, inside the module but not within any procedure internal to that module.

In a class or structure, the category of a nonlocal variable depends on whether it is shared. If it is declared with the Shared keyword, it exists in a single copy shared among all instances of the class or structure. Otherwise it is an instance variable, and a separate copy of it is created for and available only to each instance of the class or structure.

Declaring Data Types

The As clause in a declaration statement allows the developer to define the data type or object type of the variable he or she is declaring. Any of the following types can be specified for a variable:

- An elementary data type, such as Boolean, Long, or Decimal
- A composite data type, such as an array or structure
- An object type or class, such as a TextBox or DataSet

Variables can be declared in one statement without the need to repeat the data type. All the following are valid variable declarations:

```
Dim I as Integer                    'An Integer variable
Dim I, J, K As Integer              'All three are Integer variables
Dim s As Long, y As Single's is Long, y is Single.
```

Declaring Scope

A variable's scope is determined by where that variable is declared. Code located in a given region can use the variables defined in that region without having to qualify their names. In declaring scope, the following rules apply:

- The scope of a module variable is the entire namespace in which the module is defined.
- The scope of a shared or instance variable is the class in which it is declared.
- The scope of a local variable is the procedure in which it is declared. If, however, one declares a variable within a block, its scope is that block only. A block is a set of statements terminated by an End, Else, Loop, or Next statement; for example, a For…Next or If...Then...Else...End If construction.

Declaring Lifetime

The lifetime of a variable is the period of time during which it is available for use. A local variable declared with a Dim statement exists only as long as its procedure is executing. However, if it is declared with the Static keyword, a local variable continues to exist and preserve its value even when the procedure ends. Module, shared, and instance variables retain their values as long as the application runs.

Declaring Accessibility

A variable's accessibility is determined by which keyword or keywords (Dim, Public, Protected, Friend, Protected Friend, or Private) is used in the declaration statement. A module, structure, class, or instance variable can be declared with any of these keywords. Within a procedure, only the Dim keyword is allowed, and the accessibility is always private.

Object Variable Declaration

Object variables are declared in a manner similar to that used to declare other variables except the data type, which is specified as an object or a specific class from which an object can be instantiated. When a variable is declared with a specific object class, it can access all the methods and properties exposed by that class. One also can use Protected, Friend, Protected Friend, Private, Shared, or Static for the declaration. The following syntax declares an object variable:

```
Dim obj As (Object type)
```

Variable Assignment

Variables may be assigned values, consistent with their data type, at declaration or later in code. Assignment statements perform calculations and assign the result to a variable. For example:

```
Dim I as Integer = 10      'Variable I declared and assigned
I = I + 2                  'I is assigned value after declaration
```

In addition to storing values, a variable can refer to an object. Once the developer assigns an object to the variable, the developer can treat it exactly the same way he or she treats the object to which it refers. A normal assignment statement is used to assign an object to an object variable. One can assign an object expression or the Nothing keyword, as in the following code:

```
obj = ObjectClass 'Assign an object reference
```

Assignment and declaration can be combined by using the New keyword. The following declaration statement declares an object variable and assigns a specific object to it:

```
Dim ds As New DataSet  'Create and assign
```

An object variable can refer to data of any type. The value that is stored in an object variable is kept elsewhere in memory, while the variable itself holds a pointer to the data. Visual Basic supplies functions that return information about what an object variable refers to, as shown in Table A.1.

Table A.1: Functions to Interrogate Variable Value Types

FUNCTION	RETURNS TRUE IF THE OBJECT VARIABLE REFERS TO
IsArray	An array of values rather than a single value
IsDate	A number or string that can be interpreted as a date and time value
IsNumeric	A number or something that can be interpreted as a number
TypeOf...Is	The specific object Is refers to or a class in the object's inheritance hierarchy

Data Types

The data type of a programming element refers to the kind of data it can hold and how those data are stored. Data types apply to all values that can be stored in computer memory or participate in the evaluation of an expression. Every variable—literal, constant, property, procedure argument, and procedure value—has a data type.

Table A.2 shows the Visual Basic .NET data types, their supporting common language runtime types, their storage requirements, and their value ranges.

Table A.2: Data Type Summary

VISUAL BASIC TYPE	COMMON LANGUAGE RUNTIME TYPE	STORAGE	VALUE RANGE
Boolean	System.Boolean	2 bytes	True or False

Byte	System.Byte	1 byte	0 through 255 (unsigned)
Char	System.Char	2 bytes	0 through 65535 (unsigned)
Date	System.DateTime	8 bytes	Jan 1, 0001 through Dec 31, 9999
Decimal	System.Decimal	16 bytes	Up to +/− 79,228E24
Double	System.Double	8 bytes	−1.79769313486231570E + 308 through 1.79769313486231570E + 308
Integer	System.Int32	4 bytes	−2,147,483,648 through 2,147,483,647
Long	System.Int64	8 bytes	−9,223,372,036,854,775,808 through 9,223,372,036,854,775,807
Object	System.Object	4 bytes	Any type can be stored as type Object
Short	System.Int16	2 bytes	−32,768 through 32,767
Single	System.Single	4 bytes	−3.4028235E+38 to 3.4028235E+38
String	System.String	2 bytes per char	Up to approximately 2 billion Unicode characters
Structure	System.ValueType	Depends on structure definition	Each member of structure has a range determined by its data type independent of the ranges of other members

In addition to the elementary data types Visual Basic supplies, items of different types, value types, or other composite types can be assembled to create composite data types such as structures, arrays, and classes. For example, one can define an array of structure elements, or a structure with array members.

Arrays allow reference to a series of variables by the same name and unique index number. Arrays do not have fixed size in Visual Basic. One can change the size of an array after one creates it. The ReDim statement assigns a completely new array object to the specified array variable. Therefore, ReDim can change the length of each dimension.

Array variables are declared in the same manner as other variables, using the Dim statement. In the declaration, add one or more pairs of parentheses after the variable name to define its dimensions. For example:

```
Dim BA() As Byte = New Byte(2)
```

When an array is "ReDimmed," its existing values normally are lost. However, including the Preserve keyword in the ReDim statement retains those values. For example, the following statement allocates a new array, initializes its elements from the corresponding elements of the existing MyArray, and assigns the new array to MyArray.

```
ReDim Preserve MyArray(10, 20)
```

Operators and Order of Precedence

When expressions contain operators from more than one category, they are evaluated according to the following rules. The arithmetic and concatenation operators have an order of precedence that is described here, and all have higher precedence than do the comparison and logical operators. Comparison operators in turn have higher precedence than do logical operators. All comparison operators have equal precedence; that is, they are evaluated from left to right. Arithmetic, concatenation, and logical operators are evaluated in the order of precedence indicated in Table

Table A.3: Operator Precedence Summary

ARITHMETIC	COMPARISON	LOGICAL
Exponentiation (^)	Equality (=)	Negation (Not)
Negation (−)	Inequality (<>)	Conjunction (And, AndAlso)
Multiply and divide (*, /)	Less than, greater than (<, >)	Disjunction (Or, OrElse)
Modulus (Mod)	Greater than or equal to (>=)	
Add and subtract (+, −)	Less than or equal to (<=)	
String concatenation (&)	Like	
	Is	
	TypeOf...Is	

Parentheses override the order of precedence and force some parts of an expression to be evaluated before others are. Operations within parentheses are always performed before outside operations. Within parentheses, however, operator precedence is maintained.

Control Flow

Control−flow statements permit the regulation of the flow of a program's execution. By using control statements, a developer can write Visual Basic code that makes decisions with If...Then and Select...Case statements and repeat actions with Do and For loops.

If...Then...Else Statements

If...Then...Else statements are used to execute statements, depending on the true or false value of a Boolean condition. The condition usually results from a comparison of two values, but it can be any expression that evaluates to a Boolean value (True or False).

To execute only one statement when a condition is True, one can use the single−line syntax of If...Then...Else, omitting the Else and End If statements, as in the following example:

```
If condition Then code statement
```

To execute more than one line of code when the condition is True, use multiple−line syntax, which includes the End If statement. One can use the If...Then...Else statement to define two blocks of executable statements. One block is executed if the condition is True, and the other if it is False.

```
If condition Then
  code statements
Else
  code statements
End If
```

Additionally, one can add one or more ElseIf statements to If...Then...Else to test additional conditions if the first condition is False. Any number of ElseIf statements can be included, or a developer can include none at all. One also can omit an Else statement regardless of whether one has ElseIf statements. The following is a complete example:

```
If condition Then
   code statements
ElseIf condition Then
   code statements
Else
   code statements
End If
```

If...Then...Else statements can be nested to as many levels as needed. However, for readability, one might want to use Select...Case statements rather than multiple levels of nested If...Then...Else statements or many ElseIf statements.

Select...Case Statements

Visual Basic compares the value of the expression to the values in the Case statements in the order in which they appear in the Select...Case block. If it finds a match or a Case Else statement, it executes the corresponding statement block. In any case, it then executes the code that follows the End Select statement. One can have any number of Case statements and can include or omit a Case Else statement.

In the following example, Select...Case is used to evaluate an expression. Note that each Case statement can contain more than one value, a range of values, or a combination of values and comparison operators. When a Case statement contains more than one value, the Case block is executed if any of the values matches the value of the Select statement expression:

```
   Select expression
     Case 1   ' expression is 1
code statements
     Case 2, 3, 4   ' expression is 2, 3, or 4
code statements
     Case 5 To 7   ' expression is 5, 6, or 7
      code statements
     Case Is < 15   ' expression is between 8 and 14
      code statements
     Case Else   ' expression is anything else
      code statements
   End Select
```

Loop Structures

Loop structures allow a developer to execute one or more lines of code repetitively. One can repeat the statements until a condition is true, until a condition is false, a specified number of times, or once for each object in a collection. The loop structures supported by Visual Basic include While, Do...Loop, For...Next, and For Each...Next.

The While statement executes a block of statements an indefinite number of times, depending on the True or False value of a Boolean condition. The statements are repeated as long as the condition is True. The While statement always checks the condition before it begins the loop. Looping continues while the condition remains True. Its syntax is as follows:

```
  While condition
     code statements
  End While
```

Do...Loop Statements

Do...Loop statements behave similarly to While statements but permit considerably more functionality. The condition may be checked before entering the loop or at the end of the loop. Looping continues while the condition remains True. To test a condition before the loop commences, use the following syntax. Note that if the condition initially evaluates to False, the code statements will not execute:

```
Do While condition
   code statements
Loop
```

To assure that the looping code statements execute at least once, use the Do...Loop While structure. This variation of Do...Loop evaluates the condition at the end of the code statements and is implemented with the following syntax:

```
Do
   code statements
Loop While condition
```

Additionally, the Do...Loop permits substituting the word Until for While to the loop while the condition is False. For example, the Do While loop above could be modified as follows:

```
Do Until condition
   code statements
Loop
```

This variation causes the iterations to continue until the condition becomes True rather than looping while the condition is True.

For...Next Statements

Do loops work well when one does not know in advance how many times one has to execute the statements in the loop. However, when a developer expects to execute the loop a specific number of times, a For...Next loop is a better choice. Unlike a Do loop, a For loop uses a variable called a *counter* that increases or decreases in value during each repetition of the loop. If the value is omitted, it is taken to be 1. The syntax is as follows:

```
For counter = start To end [ Step step ]
   ' Statement block to be executed for each value of counter
Next [ counter ]
```

When execution of the For...Next loop begins, Visual Basic evaluates *start, end,* and *step.* It then assigns *start* to *counter.* Before it executes the statement block, it compares *counter* to *end.* If *counter* is already past the end value, the For loop terminates and control passes to the statement that follows the Next statement. Otherwise the statement block is executed. The Step parameter, which specifies the value the counter is incremented on each loop, is optional and defaults to 1.

For Each...Next Statements

The For Each...Next loop is similar to the For...Next loop, but it executes the statement block for each element in a collection instead of a specified number of times. The syntax is as follows:

```
For Each elementvariable In collection
   code statements for each value
```

```
Next [ elementvariable ]
```

For each iteration of the loop, Visual Basic sets the variable *elementvariable* to one of the elements in the collection and executes the statement block. When all the elements in the collection have been assigned to *elementvariable,* the For Each loop terminates and control passes to the statement that follows the Next statement.

Object–Oriented Programming

Almost everything that is done in Visual Basic is associated with objects. The words class and object are used so much in object–oriented programming that it is easy to mix up these terms. Generally, a class is an abstract representation of something, whereas an object is a usable example of the thing the class represents.

Classes include fields, properties, methods, and events. Fields and properties represent information that an object contains. Fields are like variables in that they can be read or set directly. Properties are retrieved and set like fields but are implemented by using Property Get and Property Set procedures, which provide more control over how values are set or returned.

Methods represent actions that an object can perform. Methods can be defined by adding procedures, either subroutines or functions, to the class. Events are notifications an object receives from or transmits to other objects or applications. Events allow objects to perform actions whenever a specific occurrence takes place. Because Microsoft Windows is an event–driven operating system, events can come from other objects, applications, or user input, such as mouse clicks and key presses.

Encapsulation, Inheritance, and Polymorphism

Fields, properties, methods, and events constitute only half of the object–oriented programming equation. True object–oriented programming requires objects to support three qualities: encapsulation, inheritance, and polymorphism.

Encapsulation means that a group of related properties, methods, and other members are treated as a single unit or object. Objects can control how properties are changed and methods are executed. For example, an object can validate values before allowing property changes. Encapsulation also hides the implementation detail from the clients using the object.

Inheritance describes the ability to create new classes that are based on an existing class. The new class inherits all the properties, methods, and events of the base class and can be customized with additional properties and methods.

Polymorphism means that it is possible to have multiple classes that can be used interchangeably even though each class implements the same properties or methods in different ways. Polymorphism is essential to object–oriented programming because it allows the use of items with the same names no matter what type of object is in use at the moment.

Overloading, Overriding, and Shadowing

Overloading, overriding, and shadowing are similar concepts that are easy to confuse. Although all three techniques allow a developer to create members with the same name, there are some important differences:

- *Overloaded members* provide different versions of a property or method that have the same name but that accept a different number of parameters or parameters with different data types.
- *Overridden properties* and methods replace an inherited property or method that is not appropriate in a derived class. Overridden members must accept the same data type and number of arguments. Derived classes inherit overridden members.
- *Shadowed members* locally replace a member that has broader scope. Any type can shadow any other type. For example, one can declare a property that shadows an inherited method with the same name. Shadowed members cannot be inherited.

What Is an Object?

An object is a combination of code and data that can be treated as a unit. An object can be a piece of an application such as a control or a form. An entire application also can be an object.

In Visual Basic .NET a class defines each object. Objects are created as identical copies of their class. Once they exist as individual objects, however, their properties can be changed and each becomes its own unique instance of the class.

Objects can be related to each other in several ways. One of these relationships is hierarchical. Classes that are derived from more fundamental classes are said to have a hierarchical relationship. Class hierarchies are useful in describing items that are a subtype of a more general class. Derived classes inherit members from the class they are based on, allowing the developer to add complexity as he or she progresses in a class hierarchy.

Another way in which objects are related is the object–container relationship. Container objects encapsulate other objects. For example, an instance of class with a property that stores an object is a container object. One type of object containment is represented by collections. Collections are like arrays of objects that can be enumerated. Visual Basic .NET supports a specific syntax in the form of For Each...Next syntax blocks that allows a developer to iterate the items in a collection.

Almost everything that is done in Visual Basic .NET involves objects. Objects and their properties, methods, and events are the basic units of object–oriented programming. An object is an element of an application that represents an instance of a class. Properties, methods, and events are the building blocks of objects and constitute their members.

An object represents an instance of a class, such as Form, Control, or Component. In Visual Basic code, it is necessary to instantiate an object before applying one of the object's methods or changing the value of one of its properties. *Instantiation* is the process by which an instance of a class is created and assigned to an object variable.

A *property* is an attribute of an object that defines one of the object's characteristics, such as size, color, or screen location, or an aspect of its behavior, such as whether it is enabled or visible. All interactions with properties are done either to read its value or to change it.

A *method* is an action that an object can perform. For example, collections have an add method that adds a new member to the collection.

An *event* is an action recognized by an object, such as clicking the mouse or pressing a key, and for which one can write code to respond. Events can be triggered by user action, program code, or the system. A component developer also can create custom events to be raised by the objects and handled by other objects.

Delegates and Events

Events and delegates are closely associated in Visual Basic .NET. An *event* is a message sent by an object announcing that something important has happened. Events are implemented by using *delegates,* a form of object–oriented function pointer that allows a function to be invoked indirectly by means of a reference to the function.

Events and Event Handlers

Most modern programs are event–driven; that is, the flow of execution is determined by external occurrences or events initiated the user, the program code, or the system. An event is a message that informs an application that something has happened. For example, when a user clicks a control on a form, the form can raise a Click event and call a procedure that performs the action the developer desires when that particular form is clicked.

Declaring and Raising Events

Many objects that are available to a Visual Basic developer can generate events. Developers also can declare their own events within classes, structures, modules, and interfaces by using the Event keyword, as in the following example:

```
Event AnEvent(ByVal EventNumber As Integer)
```

Once an event is declared within a class, structure, and so forth, it must be raised to send the event message. The act of broadcasting the message is called *raising* the event. In Visual Basic .NET events are raised with the RaiseEvent statement, as in the following example:

```
RaiseEvent AnEvent(EventNumber)
```

Any object capable of raising an event is an event sender, also known as an event source. Forms, controls, and user–defined objects are examples of event senders.

Event handlers are procedures that are called when a corresponding event occurs. One can use any valid subroutine as an event handler. Functions cannot serve as event handlers because they cannot return a value to the event source.

Visual Basic uses a standard naming convention for event handlers that combine the name of the event sender, an underscore, and the name of the event. For example, the click event of a button named button1 would be named Sub button1_Click.

Associating Events with Event Handlers

Before an event handler becomes usable, one first must associate it with an event by using either the Handles or the AddHandler statement.

The WithEvents statement and Handles clause provide a declarative way to specify event handlers. Events raised by an object declared with WithEvents can be handled by any subroutine with a Handles clause that names an event. Although the Handles clause is the standard way of associating an event with an event handler, it is limited to associating events with event handlers at compile time.

The AddHandler and RemoveHandler statements are more flexible than the Handles clause. They allow the dynamic connection and disconnection of events with one or more event handlers at runtime, and they do not require the developer to declare object variables by using WithEvents.

Delegates and the AddressOf Operator

Delegates are objects that call the methods of other objects. They sometimes are described as type–safe function pointers because they are similar to the function pointers used in other programming languages. However, unlike function pointers, Visual Basic .NET delegates are a reference type based on the class System.Delegate and can reference both shared and instance methods.

Delegates are useful in situations in which one needs an intermediary between a calling procedure and the procedure being called. For example, a developer might want an object that raises events to be able to call different event handlers under different circumstances. Unfortunately, the object–raising events cannot know ahead of time which event handler is handling a specific event. Visual Basic .NET lets a developer dynamically associate event handlers with events by creating a delegate for the developer when he or she uses the AddHandler statement. At runtime, the delegate forwards calls to the appropriate event handler.

Although it is possible to create one's own delegates, in most cases Visual Basic .NET creates the delegate and takes care of the details for the developer. For example, an Event statement implicitly defines a delegate class named (EventName)EventHandler as a nested class of the class containing the Event statement with the same signature as the event. The AddressOf statement implicitly creates an instance of a delegate.

In some situations it may be desirable to declare an event by using an existing delegate type as its underlying delegate. This is useful when one wants to route multiple events to the same handler. The following syntax demonstrates how this is done:

```
Event AnEvent As DelegateType
```

Error Handling

Visual Basic supports structured exception (error) handling, which allows a program to detect and possibly recover from errors during execution. Visual Basic uses an enhanced version of the Try...Catch...Finally syntax that is already supported by other languages, such as C++. Structured exception handling combines a modern control structure (similar to Select...Case or Do...Loops) with exceptions, protected blocks of code, and filters.

Structured exception handling, which is the recommended method for error handling in Visual Basic, makes it easy to create and maintain programs with robust, comprehensive error handlers. Unstructured exception handling using On Error can degrade application performance and result in code that is difficult to debug and maintain.

Introduction to Exception Handling

Visual Basic supports both structured and unstructured exception (error) handling. By placing specific code in an application, a developer can handle most of the errors users may encounter and enable the application to continue running. Structured and unstructured error handling allows a developer to plan for potential errors and prevent them from interfering with the purpose of the

application. Consider using exception handling in any method that uses operators that may generate an exception or that calls into or accesses other procedures that may generate an exception.

If an exception occurs in a method that is not equipped to handle it, the exception is propagated back to the calling method. If the previous method also has no exception handler, the exception is propagated back to that method's caller, and so on. The search for a handler continues up the call stack, which is the series of procedures called within the application. If the search fails to find a handler for the exception, an error message is displayed, and the application is terminated.

In structured exception handling, blocks of code are encapsulated, with each block having one or more associated handlers. Each handler specifies some form of filter condition for the type of exception it handles. When an exception is raised by code in a protected block, the set of corresponding handlers is searched in order, and the first one with a matching filter condition is executed. A single method can have multiple structured exception–handling blocks, and the blocks also can be nested. The Try...Catch...Finally statement is used specifically for structured exception handling.

The On Error statement is used specifically for unstructured exception handling. In unstructured exception handling, On Error is placed at the beginning of a block of code. It then has scope over that block; it handles any errors occurring within that block. If the program encounters another On Error statement, that statement becomes valid, and the first statement becomes invalid.

In Visual Basic errors (also called exceptions) fall into one of three categories: syntax errors, runtime errors, and logic errors.

- *Syntax errors* appear while a developer writes the code. Visual Basic checks the code as the developer types and alerts the developer if he or she makes a mistake, such as misspelling a word or using a language element improperly. Syntax errors are the most common type of errors. One can fix them easily in the coding environment as soon as they occur.
- *Runtime errors* appear only after the developer compiles and runs the code. They involve code that may appear to be correct in that it has no syntax errors but that will not execute. For example, one might attempt to open a file, but if the file does not exist, the application cannot carry out the function.
- *Logic errors* appear after an application is in use. They most often take the form of unwanted or unexpected results in response to user input or simply faulty application logic. Logic errors are generally the hardest type to fix, since it is not always clear where they originate.

Structured Exception Handling

Visual Basic supports structured exception handling, which helps a developer create and maintain programs with robust, comprehensive error handlers. Structured exception handling is code designed to detect and respond to errors during execution by combining a control structure with exceptions, protected blocks of code, and filters.

By using the Try...Catch...Finally statement, it is possible to protect blocks of code that have the potential to raise errors. One can nest exception handlers; the variables declared in each block have local scope.

The following syntax shows the structure of a Try...Catch...Finally statement:

```
Try  '  Starts a structured exception handler
    ' code statements that may generate exceptions
```

```
Catch [optional filters]
    ' code statements in response to filtered exception
Catch [additional filters]
    ' code statements in response to filtered exception
Finally
    ' codes statements before the Try statement exits
End Try    ' Ends a structured exception handler
```

The Try block of a Try...Catch...Finally exception handler contains the section of code the developer wants the error handler to monitor. If an error occurs during execution of any of the code in this section, Visual Basic examines each Catch statement within the Try...Catch...Finally block until it finds one whose condition matches that error. If one is found, control transfers to the first line of code in the Catch block. If no matching Catch statement is found, the search proceeds to the Catch statements of the outer Try...Catch...Finally block that contains the block in which the exception occurred. This process continues through the entire stack until a matching Catch block is found in the current procedure. If none is found, an error is produced.

The code in the Finally section always executes last, just before the error-handling block loses scope, regardless of whether the code in the Catch blocks has executed. Place cleanup code, such as that for closing files and releasing objects, in the Finally section.

Catch blocks allow three options for specific error filtering. In one, errors are filtered on the basis of the class of the exception (in this case ClassLoadException). If a ClassLoadException error occurs, the code within the specified Catch block is executed:

```
Try
    ' "Try" block code
Catch e as ClassLoadException
    ' "Catch" block code
Finally
    ' "Finally" block code
End Try
```

In the second error-filtering option, the Catch section can filter any conditional expression. A common use of this form of Catch filter is to test for specific error numbers, as shown in the following code:

```
Try
    ' "Try" block code
Catch When ErrNum = 5 'Type mismatch
    ' "Catch" block code
Finally
    ' "Finally" block code
End Try
```

When Visual Basic finds the matching error handler, it executes the code within that handler and then passes control to the Finally block.

As a third alternative, one can combine both options for exception handling.

In trying to find a Catch block to handle an exception, each block's handler is evaluated until a match is found. Since these handlers can be calls to functions, unexpected side effects may result; for example, such a call might change a public variable that then is used in the code of a different Catch block that ends up handling the exception.

Unstructured Exception Handling

In unstructured exception handling, an On Error statement is placed at the beginning of a block of code, and that statement handles any errors occurring within that block. When an exception is raised in a procedure after the On Error statement executes, the program branches to the line argument specified in the On Error statement. The line argument, which is a line number or line label, indicates the location of the exception handler.

Sometimes a call is made from the original procedure to another procedure, and an exception occurs in the called procedure. In such cases, if the called procedure does not handle the exception, the exception propagates back to the calling procedure and execution branches to the line argument.

Unstructured error handling uses three error–handling statements: On Error GoTo Line, On Error Resume Next, and On Error GoTo 0.

On Error GoTo Line

The On Error GoTo Line statement assumes that error–handling code starts at the line specified in the required *line* argument. If a runtime error occurs, control branches to the line label or line number specified in the argument, activating the error handler. The specified line must be in the same procedure as the On Error GoTo Line statement; otherwise, Visual Basic generates a compiler error. The following example illustrates the use of an error handler with a line label:

```
Sub TestSub
On Error GoTo ErrorHandler
     ' Code that may contain errors
   Exit Sub
ErrorHandler:
     ' Code that handles errors
     Resume
End Sub
```

The example contains an error handler named ErrorHandler. If any code in the TestSub subroutine generates an error, Visual Basic immediately executes the code that follows the ErrorHandler label. At the end of the error–handling block the Resume statement passes control back to the line of code where the error first occurred. The rest of the subroutine then continues executing as if the error did not occur.

The developer must place an Exit Sub statement immediately before the error–handling block. Otherwise, Visual Basic runs the error–handling code when it reaches the end of the subroutine, causing unwanted or unexpected results.

On Error Resume Next

The On Error Resume Next statement specifies that if there is a runtime error, control passes to the statement that immediately follows the one in which the error occurred. At that point, execution continues. On Error Resume Next enables a developer to put error–handling routines where errors will occur rather than transfer control to another location in the procedure.

If the procedure calls another procedure, the On Error Resume Next statement becomes inactive during the execution of the called procedure. Therefore, the developer should place an error–handling statement in each called procedure that needs one. This is necessary because the Resume Next behavior applies only to the procedure containing the On Error Resume Next

statement. If an unhandled error occurs in a called procedure, the exception propagates back to the calling procedure, and execution resumes on the statement that follows the call. In such cases the error is not handled.

Resume also can be used on its own, outside the On Error statement. When Resume is used this way, Visual Basic returns control to the statement that caused the error. One generally uses Resume after an error handler corrects the error.

Visual Basic also provides the Resume Next statement, which directs control to the line that immediately follows the line of code that caused the error. A developer might use Resume Next for cases in which an error will not cause the application to stop working. One also might use it if an error will not change the expected results of the subroutine.

On Error GoTo 0

The On Error GoTo 0 statement disables any error handler in the current procedure. If a developer does not include an On Error GoTo 0 statement, the error handler is still disabled when the procedure containing the exception handler ends.

Appendix B: The Role of Visual Studio .NET

Overview

Visual Studio .NET is the newest entry in the wave of development products released by Microsoft. Visual Studio .NET was created with the following design goals:

- Build the next–generation Internet
- Create powerful applications fast and effectively
- Span any platform or device

Visual Studio .NET is the only development environment built from the ground up for XML Web services. By allowing applications to share data over the Internet, XML Web services enables developers to assemble applications from new and existing code regardless of the platform, programming language, or object model.

Visual Studio .NET is available in the Professional, Enterprise Developer, and Enterprise Architect editions. This appendix outlines these three editions and the features that are included with each one. Through the process of defining the components that make up the different editions, the reader will see the role of Visual Studio .NET and have the information he or she needs to decide which edition is right for his or her organization.

The appendix starts with the Professional Edition and describes the products that are included with that release. It then introduces the Enterprise Developer Edition and describes the additional components that are included with that release over and above the Professional Edition. Finally, the appendix outlines the features that are included with only the Enterprise Architect edition. For more information on each of the developer tools, visit Microsoft's Web site at http://www.microsoft.com/ and check out the Visual Studio homepage.

Visual Studio .NET Professional Edition

Visual Studio .NET enables a developer to build applications that integrate across programming languages and can target any Internet device. One can build and deploy XML Web services and applications rapidly. The following sections categorize the features into the following areas:

- Applications
- Languages
- Tools and environments
- Database applications
- Enterprise tools
- Server technologies

Applications

The Professional Edition of Visual Studio .NET is full of application options for developers. The following list outlines the types of applications that can be created by using the Professional Edition:

- **XML Web services.** XML Web services can be used and created from any of the Visual Studio programming languages. A developer can expose any component as an XML Web service, making it accessible from almost any location. The developer uses the WebMethod

attribute to expose the components.

- **Windows Forms.** Windows Forms is the forms designer that helps a developer build Windows–based applications. A developer can take advantage of features such as form inheritance, control anchoring, and docking enable.
- **Web Forms.** Web Forms is a Web page editor that allows a developer to create Web applications by using the same code techniques that have helped make Microsoft Visual Basic one of the most widely used development tools in the world. One can take advantage of drag–and–drop and double–click code to simplify the development process and provide assistance in creating fully functional Web applications. Web Forms can be created in any language, including Visual Basic .NET and C# .NET.
- **.NET Framework and common language runtime.** Built for the Web, the .NET Framework provides an agile, scalable environment for building, deploying, and running distributed XML Web services and applications.
- **Windows services.** Windows services allows a developer to build long–running executable applications that can be started automatically at boot time and run in his or her own Windows sessions.
- **Windows control libraries.** This feature allows a developer to construct custom user interface controls for Windows Forms by using the Windows control library template.
- **Web control libraries.** This feature allows a developer to construct custom Web server controls by using the Web control library template.
- **Class libraries.** This feature allows a developer to create reusable class library components by using the class library template.
- **Console applications.** This feature allows a developer to build text–based applications for Microsoft .NET that execute within the console window.

Languages

Visual Studio .NET ships with several Microsoft development languages, which are provided to help a developer create the types of applications referred to in the previous section. The following list identifies the languages included in the Professional Edition:

- **Visual Basic .NET.** Visual Basic .NET enables full object–oriented programming with implementation inheritance, structured exception handling, and free threading.
- **Visual C# .NET.** Visual C# .NET, a new object–oriented programming language, combines the power of C and C++ with the functional ease of modern, rapid application development (RAD) tools.
- **Visual C++ .NET.** Microsoft Visual C++ .NET enables developers to build managed and unmanaged applications by using the .NET Framework, Active Template Library (ATL) Server and Microsoft Foundation Classes (MFC).
- **Visual J# .NET.** Microsoft Visual J# .NET is a development tool for Java–language developers who want to build applications and services on the .NET Framework.
- **Built–in support for additional programming languages.** With support for multiple programming languages, a developer can program in the language of his or her choice and integrate the code from any other language.

Development Tools and Environments

Visual Studio .NET has set the industry standard for tools and work environments. The following tools are available with the Professional Edition to facilitate the overall development process:

- **RAD for the server.** Accesses and integrates server administration tools, event logs, databases, and XML Web services. Server Explorer and tools incorporate server–side

application components.

- **Visual Studio .NET Debugger.** Shortens the development cycle through the use of cross–language, integrated debugging.
- **Visual Studio .NET integrated development environment (IDE).** Leverages a tightly integrated visual development environment for all languages; includes a single toolbox, task list, and debugger.
- **Dynamic Help.** Instantly accesses help that is relevant to the current development task.
- **Task List.** Organizes the development process by annotating code and monitoring and quickly accessing errors and warnings.
- **HTML Designer.** Develops HTML, Active Server Pages (ASP), and Microsoft ASP.NET Web forms graphically without the need to write HTML or script.
- **Crystal Reports.** Builds rich interactive reports for Windows–based, Web–based, and mobile applications as well as Web services.

Database Applications

Until this point in the appendix, all the editions of Visual Studio .NET have been identical. All three editions give the same support for the applications, languages, and tools mentioned in the previous sections.

This is where the differences start to show. The Professional Edition of Visual Studio does include some of the database applications, but the other two editions extend the options. The following items are included as database applications with the Professional Edition:

- **Microsoft SQL Server 2000 Desktop Engine (MSDE 2000).** Builds applications that are fully compatible with SQL Server and migrates them directly to SQL Server without changing a single line of code. However, this is not the full version of SQL Server 2000.
- **Visual database tools.** Design stored procedures, indexes, tables, triggers, user–defined functions, and other database elements visually. In the Visual Studio .NET Professional Edition, these design capabilities are limited to the SQL Server Desktop Engine.
- **XML Designer.** Uses drag–and–drop tools for working with XML and XML Schema Definition (XSD) files.

To take advantage of these two categories, one has to step up to one of the other editions of Visual Studio. More information about the other two editions is given in the following sections.

Visual Studio .NET Enterprise Developer Edition

The Enterprise Developer Edition includes several features that facilitate enterprise development. This edition includes everything the Professional Edition does. The following sections outline the additional features included with the Enterprise Developer Edition.

This section starts with the additional database applications mentioned in the preceding section. Then it introduces the enterprise tools included with this edition and concludes with an overview of the server technologies that are included.

Database Applications

The Enterprise Developer Edition of Visual Studio .NET has one primary enhancement in the database applications category: The full version of SQL Server 2000 is included with this edition. (It

should be noted that the Enterprise Architect edition of Visual Studio also includes SQL Server 2000.) For more information on the features and configuration of SQL Server 2000, refer to Chapter 8, "SQL Server 2000."

Enterprise Tools

The following Enterprise Tools are included with the Enterprise Developer edition:

- **Microsoft Visual SourceSafe 6.0c.** Provides a collaborative team development environment with version control for source code, components, applications, and design documents.
- **Application Center Test.** Performs functional, performance, and load testing of XML Web services and applications and automates regression tests.
- **Enterprise templates and frameworks.** Take advantage of application templates to jump–start development and access best practices with the XML–based Template Description Language.
- **Microsoft .NET–based reference applications.** Show how to design and structure .NET–based applications with end–to–end reference applications, including design documents, models, and code.
- **Visual Studio Analyzer.** Locates performance bottlenecks visually in distributed COM applications.

Server Technologies

The final category of features within Visual Studio can be one of the most important for Enterprise developers. The Enterprise Architect and Enterprise Developer editions both include the following products. The products are included with these editions of Visual Studio .NET but are licensed for test and development purposes only. Those who have questions related to licensing should refer to the product documentation or installation guide of the product in question. Most of these products have a chapter in this book dedicated to them:

- Windows 2000 Standard Server
- SQL Server 2000
- Exchange Server
- Commerce Server
- Host Integration Server
- BizTalk Server

Visual Studio .NET Enterprise Architect Edition

The Enterprise Architect Edition of Visual Studio is the fullest–featured of all the editions. It comes with all the features and components that have been described earlier in this appendix for the other editions. The added components are in the categories of database applications and Enterprise tools, as discussed next.

Database Applications

The Enterprise Architect Edition includes a single add–on to the previously outlined features. A Microsoft Visio–based database modeling tool is provided that allows a developer to capture and communicate business requirements clearly with conceptual, logical, and physical database

modeling tools.

Enterprise Tools

The Enterprise tools category is where the largest number of features have been added to the Enterprise Architect Edition, including the following tools:

- **Visio−based UML application modeling.** Models application architecture and functionality visually and clearly communicates requirements with a development team.
- **Enterprise template project type.** Shares best practices and provides architectural guidance across the development team with the new Enterprise template project type.

Summary

Visual Studio .NET consists of the set of tools provided to a developer to create applications. The set of tools is comprehensive and is intended to assist in the development of the applications a developer needs in a timely and efficient manner.

The three editions are provided to allow developers to scale in the areas where they need to. For many developers the set of features, components, and tools provided with the Professional Edition is more than they could possibly use. For others the data−modeling tools available only with the Enterprise Architect Edition are worth their weight in gold. Developers will need to find the options that best suit them and their organizations. Clearly, the tools provided with Visual Studio .NET are a complete development solution to interact with the .NET Enterprise Servers described in this book.

List of Figures

Chapter 1: Introduction to the .NET Framework

Chapter 2: .NET Business Solutions

Chapter 3: Role of .NET Enterprise Servers

Chapter 4: .NET Services for the Development Environment

Chapter 5: XML Integration with .NET Servers

Chapter 6: Building .NET Components

Chapter 11: Content Management Server

Chapter 12: BizTalk Server

Chapter 13: SharePoint Portal Server

Chapter 14: Internet Security and Acceleration Server

Chapter 15: Web Services with .NET

Chapter 16: Building Enterprise Web Farms

Chapter 17: Data Warehousing

Chapter 18: Clustering .NET Servers

List of Tables

Chapter 3: Role of .NET Enterprise Servers

Chapter 8: SQL Server 2000

Chapter 10: Commerce Server

Chapter 13: SharePoint Portal Server

Chapter 14: Internet Security and Acceleration Server

Chapter 18: Clustering .NET Servers

Appendix A: Visual Basic .NET Reference Guide