Microsoft

Operating .NET-Based Applications

patterns & practices

Version 1.0

Contents

1

Introduction

Welcome

Welcome to Operating .NET-Based Applications. This book gives you the best information available about monitoring and operating applications based on the Microsoft® .NET Framework. If you are an administrator with responsibility for application operation and monitoring, then this book is for you. However, although the primary focus is infrastructure and operations, developers involved in designing and creating applications that use the .NET Framework should find much of the information useful.

This book covers three main areas:

- Security
- Monitoring and Reporting
- Sizing and Capacity Planning (under development)

The information in this book is both practical and prescriptive. Rather than discuss all the different approaches that you could use, it concentrates on a describing a single solution that works. If you want more in-depth discussions of the concepts behind this material, refer to resources such as the .NET Framework Software Development Kit, the Visual Studio .NET documentation, and the Windows 2000 Server Resource Kit.

This book includes material from consultants working in the field and from early implementers and contains current best practice for operating .NET-based applications. We hope you enjoy reading this book and that you find the material contained within it helpful, informative, and interesting.

Who Should Read This Book

This book is aimed at administrators who are responsible for deploying and maintaining enterprise level business applications based on the Microsoft .NET Framework within hybrid data center environments. It also provides useful information for the developer writing such applications.

Prerequisites

Because of the dual audience for this book, there are different prerequisites for each group. The sections aimed at administrators assume a strong familiarity with service management procedures and a basic understanding of the principles of application development. However, these parts do not require any programming knowledge beyond basic scripting.

The sections targeted at developers assume an understanding of the development process for distributed applications and familiarity with the Microsoft Visual Studio® .NET programming tools. The sample applications are in C++, C# (C Sharp) and Visual Basic®, so development experience in these languages is an advantage. Experience with Visual Studio .NET, the .NET Framework SDK and the MSDN® Subscriptions Library will also be of benefit, but is not required in order to understand the programming sections.

Document Conventions

This guide uses the style conventions and terminology shown in Table 1.1.

Table 1.1. Document Conventions

Element	Meaning
bold font	Characters that you type exactly as shown, including commands and switches. Programming elements, such as methods, functions, data types, and data structures appear in bold font (except when part of a code sample, in which case they appear in monospace font). User interface elements are also bold.
Italic font	Variables for which you supply a specific value. For example, *Filename.ext* could refer to any valid file name for the case in question. New terminology also appears in italic on first use.
Monospace font	Code samples.
%SystemRoot%	The folder in which Windows 2000 is installed.

How to Use This Book

This book consists of three modules, each of between four and six chapters. The three modules are as follows:

- Monitoring .NET-Based Applications
- Securing .NET-Based Applications
- Capacity Planning for .NET-Based Applications (under development)

Let's look at each of these in detail.

Monitoring .NET-Based Applications

This section covers how you can monitor the functioning of your applications. This includes defining healthy and unhealthy applications and taking appropriate action based on the information you gather. This material gives you the definitive answers on how you can implement effective monitoring and notification of issues affecting .NET-based applications. The following chapters make up this section.

Chapter 2 – Monitoring Concepts

Discusses the idea of application health and distinguishes between healthy and unhealthy applications. You consider issues such as monitoring load and the importance of balancing information detail against monitoring overhead. You look at monitoring terminology and concepts that you review in later chapters.

Chapter 3 – Selecting Data Sources

The main factor in effective monitoring is collecting data about your application's health to provide you with the feedback on what is happening. In this chapter, you will look at the various sources of data that you can collect about your application's health. You will be introduced to the concepts of coarse-grained and fine-grained monitoring, which allow you to choose how much information you need to gather and how much you are willing to pay for it, in terms of either hardware or human resources.

Chapter 4 – Instrumenting .NET-Based Applications

The next part of this module looks at how you can implement application instrumentation in order to gather specific information that you cannot measure using system-provided monitors. You look at code samples that you can adapt to provide monitoring information for your own applications.

Chapter 5—Configuring Management Applications

Addresses the issues that relate to the integration of Microsoft Operations Manager (MOM), Application Center 2000 Server, and Appmetrics™ from Xtremesoft Inc. into your application environment. In particular, you look at how you can use MOM to collect the information selected in Chapters 3 and 4, and then collate this information centrally using MOM management packs.

Chapter 6—Notifying and Reporting

Shows you techniques for translating generated events either into immediate notifications or into longer-term reports. You cover what sort of information you want to collect, define your audience, and decide how you want to use the gathered data. You also look at ways in which you can deliver notifications, either immediately or on a schedule.

Securing .NET-Based Applications

Security has never had a higher profile than in the current climate and it is only appropriate that this subject receives detailed coverage. In this section, you see how to maximize the security of your environment and still allow your .NET-based applications to function properly. You cover how to use Group Policy to lock down computers and examine how to secure the network traffic generated by the application. The following chapters make up this module:

Chapter 7—General Security Recommendations

Covers the steps to ensure that your environment is secure. One of the most important steps is to make sure that your computers all run the latest patches and service packs. This chapter shows measures you can take to ensure that your IT environment remains fully up to date.

Chapter 8—Managing Security with Windows 2000 Group Policy

Windows 2000 defines many security settings through Group Policy, which controls the behavior of objects on the local computer and in the Active Directory® directory service. You must set these policies appropriately, and monitor the policies to ensure that no changes take place without prior authorization. This chapter looks in detail at managing security using Group Policy.

Chapter 9—Securing Servers Based on Role

Different server roles involved in your Active Directory environment will require different settings to make them secure. This chapter looks at domain controller and member server roles showing the steps you should take to ensure that each of these roles are as secure as possible.

Chapter 10 – Securing Servers Running .NET-Based Applications

Once you have defined overall security settings for your environment, you need to secure the specific servers running .NET-based applications. This chapter defines the settings you will require to allow your .NET-based applications to run as securely as possible.

Chapter 11 – Network Considerations for Servers Running .NET-Based Applications

Applications built on the .NET Framework can span multiple computers. In order to ensure that the applications are secure, you need to examine the communication between client and server and between servers, and then secure the traffic flow. This chapter shows you how to increase the security of network traffic flow generated by .NET-connected applications.

Chapter 12 – Customizing the Security Environment for Specific Applications

Following the general recommendations listed in the previous chapters will give you guidelines for securing any .NET-based application. However, you may need to make changes to the environment to allow your specific application to function. In this chapter, using the sample FMStocks application as an example, you see how to configure settings to allow specific applications to run.

Capacity Planning for .NET-Based Applications

Currently under development, the sizing and capacity planning section will give you the tools that you need to model the demands that an application will make before you deploy it into a production environment. This will allow you to make accurate forecasts about server scaling during concurrent code development. These chapters will also deal with performance and capacity planning, letting you set reasonable thresholds for future expansion without incurring excessive costs.

Microsoft Operations Framework (MOF)

For operations in your environment to be as efficient as possible, you should manage them effectively. To assist you, Microsoft has developed the Microsoft Operations Framework (MOF). This is essentially a collection of best practices, principles, and models providing you with operations guidance. Following MOF guidelines should help your mission critical production systems remain secure, reliable, available, supportable, and manageable.

The MOF process model consists of four integrated quadrants, as follows:

- Changing
- Operating
- Supporting
- Optimizing

Together, the phases form a spiral life cycle (see Figure 1.1) that can apply to anything from a specific application to an entire operations environment with multiple data centers. In this case, you will be using MOF in the context of monitoring and security operations.

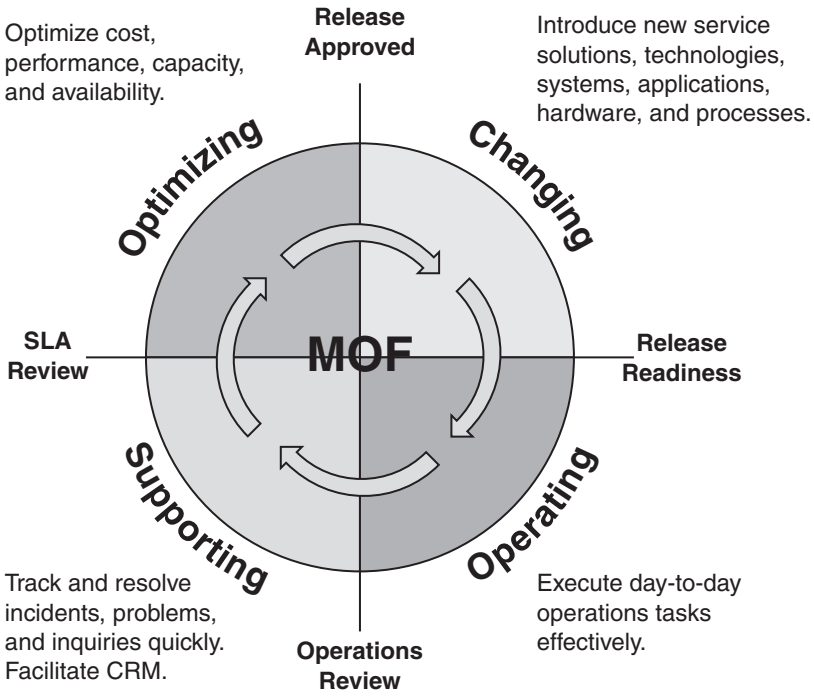


Figure 1.1
MOF lifecycle

The process model includes 20 service management functions (SMFs) and an integrated team model and risk model. Each quadrant links to a corresponding operations management review, known as a review milestone, which allows assessing of the effectiveness of that quadrant's SMFs.

It is not essential to be a MOF expert to understand and use this guide, but a good understanding of MOF principles will help you manage and maintain a reliable, available, and stable operations environment.

MOF Information

For more information about MOF, see the MOF Web Site at *http://www.microsoft.com /business/services/mcsmof.asp.*

You should also consider the Microsoft Operations Framework Executive Overview White Paper at *http://www.microsoft.com/business/services/MOFoverview.asp.*

What Are .NET-Based Applications?

Applications based on the .NET Framework fall into one or both of the following categories:

- .NET-based applications
- .NET-connected applications

.NET-based applications are programs that require the Microsoft .NET Framework to run. An example of a .NET-based application is the FMStocks scenario that this book illustrates.

.NET-connected applications use .NET-based services to communicate between the tiers in an n-tier data center design or even between organizations linked across the Internet. An example of a .NET-connected application is the Microsoft .NET Web site at www.gotdotnet.com, which uses the Microsoft Passport service for secure authentication.

Note: .NET-connected applications are effectively a sub-set of .NET-based applications. This book uses the more generic term of ".NET-based application" to encompass both .NET-based and .NET-connected applications.

You create .NET-based applications using one of the development environments that make up Visual Studio .NET. You can use these tools and the .NET Framework to develop the following types of applications and services:

- Console applications
- Scripted or hosted applications
- Windows GUI applications (Windows Forms)
- ASP.NET-based applications
- XML Web services
- Windows services

What Is Different About .NET-Based Applications?

In many ways, .NET-based applications are just like your current Windows DNA-based applications. Many of the tools, techniques and most of the experience you have gained operating your Windows DNA-based applications apply directly to operating .NET-based applications. There are also, however, several differences, which are:

- .NET-based applications run using the .NET Framework
- .NET-based applications implement Web Services
- .NET-based applications consume Web Services

You will consider the role of the .NET Framework in a moment.

Implement Web Services

Web services play an important part of the .NET application architecture and many .NET-based applications implement a Web service. Operating a .NET-based application that implements a Web service is very similar to running a program like the example FMStocks application. The difference here is that instead of generating Web pages for display in a browser, the Web service provides a programmable application component accessible through standard Web protocols. While this is a big difference for how you interact with a Web service, from an operations standpoint there is little difference in terms of running the application.

There is, however, a difference between a Web service and a Web site in terms of service levels. Web services usually have more formal Service Level Agreements (SLAs) than Web sites, so good operating practice is much more important than in a traditional Windows DNA-based application.

Consume Web Services

.NET-based applications can also act as consumers of Web services. This affects the .NET application architecture when a .NET-based application uses one or more Web services. When a .NET-based application calls a Web service in a distributed environment, it becomes a .NET-connected application.

The challenge of operating a .NET-connected application is that you must manage the SLAs of all the Web services that combine to make up the .NET-connected application. This is because the level of service provided by the .NET-connected application is the aggregate of all of the SLAs for the consumed Web services.

What Is the .NET Framework?

The .NET Framework is Microsoft's latest object-oriented programming platform that simplifies application development in highly distributed environments, including the Internet. The .NET Framework fulfills the following objectives:

- Provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- Create a code-execution environment that minimizes software deployment and versioning conflicts.
- Implement a code-execution environment that guarantees safe execution of code, including code created by an unknown or semi-trusted third party.
- Provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- Make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- Build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

Common Language Runtime

The common language runtime is the foundation of the .NET Framework. Think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remote operation. The common language runtime also enforces strict type safety and other forms of code accuracy that ensure security and robustness. Thus, it is important to know how well the common language runtime is operating as this has a direct affect on your applications.

Using the managed environment of the runtime eliminates many common software issues. For example, the runtime automatically handles object layout and manages references to objects, releasing them when no longer in use. This automatic memory management resolves the two most common application errors of memory leaks and invalid memory references.

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system (CTS). CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers generate managed code that conforms to the CTS. Therefore, managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

The common language runtime design supports a variety of different types of applications, from Web server applications to programs with a traditional rich

Windows user interface. Each type requires a runtime host to start the application. The runtime host loads the runtime into a process, creates the application domains within the process, and loads user code into the application domains.

Defining the Business Scenario

Due to the prescriptive nature of this book, the chapters cover a sample application that highlights the common scenarios. This helps you understand the best practice recommendations within this material. The sample application is the FMStocks stock trading Web site from the Visual Studio .NET Enterprise Samples. The FMStocks application provides a good balance between avoiding over-complexity and providing many of the scenarios you will encounter when operating.NET-based applications.

The assumption in this book is that you have an existing data center with one or more Windows DNA-based applications and that your company plans to roll out .NET-based applications. These applications are either new line-of-business applications or new versions of existing ones.

Much of what you do today to operate your Windows DNA applications applies directly to operating .NET-based applications. The .NET Framework allows you to use your existing knowledge, tools, and infrastructure to operate .NET-based applications. However, there will be some new concepts and tools to learn.

Fitch and Mather

Fitch and Mather is a fictitious online brokerage based in Seattle, WA. It provides a portal Web site that allows registered users from anywhere in the world to deal in stocks and shares.

Business Aims

The Fitch and Mather share-trading portal provides a simple, intuitive, fast, reliable, and secure browser-based interface that allows Internet users to carry out online research into companies and to trade in corporate stocks. Users can create an account, log on, see the current balance of their stock portfolio, view information on a potential investment, and buy and sell stock.

You can see an example of the Fitch and Mather Web site at the following Internet address:

http://www.fmstocks.com

Note: This version is Fitch and Mather 2000, which uses Windows DNA rather than the .NET Framework. However, it has exactly the same functionality as the .NET version.

Company Size and Composition

Fitch and Mather employs 150 brokers, 25 IT engineering staff, and 5 in-house developers. The brokers and the IT staff are all located at the corporate headquarters on the outskirts of Ballard, a Seattle neighborhood.

Trading Levels

Last year the company reported a turnover of $57 million, with this year on target to break the $100 million mark. This means that every minute of every day generates $190 of revenue, so an hour's downtime costs on average around $11,400 in lost earnings.

The Data Center Environment

Fitch and Mather has invested heavily in the provision of a suitable physical environment for its data center, which rivals that of many application service providers. Because of the high cost of downtime, significant investment has gone into providing the necessary physical structure to support continuous operation.

The company maintains its Web presence in a dedicated data center in a highly survivable bunker underneath its headquarters building. This building is a former civil defense command post, which Fitch and Mather bought from the City of Seattle in the mid-1990s.

The data center has standby generators and redundant utility supplies, with duplicated incoming power and water supplies at either end of the building. Two fractional T3 connections, each to different Internet Service Providers (ISPs), link Fitch and Mather to the Internet. Redundant cooling systems are more than capable of removing excessive heat and unplanned outages .

Security

With the amounts of money traded every day, Fitch and Mather need to take exceptional security precautions to guard against a range of attacks. As befits a wartime bunker, physical security is extremely high. Card access is required to move about the administrative building, but a combination of biometric retinal scanning, swipe cards, and PINs control entry to the data center itself.

The company takes a defense-in-depth approach to network security, using multiple firewalls, removing or disabling unnecessary network and computer services, and implementing strict password and security policies. Application security is also a major concern, and Fitch and Mather follow all current best practice recommendations when designing applications.

Section One of this book covers the details of the application security issues. This should be read in conjunction with Building ASP.NET applications at *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/secnetlpMSDN.asp.*

Technologies

In the seven years since its inception, Fitch and Mather has created a powerful and reliable computing infrastructure, originally based on Microsoft Windows NT® 4.0 but now upgraded to the Microsoft Windows® 2000 Server family. They use a range of technologies to provide them with the functionality that they need for their customer-facing Web site.

Although they have deployed a number of components based on the .NET Framework, Fitch and Mather still maintain and operate a number of legacy COM and COM+ applications.

Sample .NET-Based Application

The .NET-based application example that you will be using in this book is the code that runs the fictional Fitch and Mather online stock brokering Web site. This environment includes three major revisions from its original construction as a Distributed InterNetwork Application (DNA) demonstration site into today's .NET Framework implementation. The three revisions are:

- **Fitch and Mather Stocks 1.0**–Version 1.0 of the Web site used Microsoft Visual Studio 6.0 technologies, with the emphasis on Visual Basic 6.0 and the Windows NT 4.0 Server platform.
- **Fitch and Mather Stocks 2000**–The second iteration moved the original infrastructure onto the Windows 2000 platform. Additionally, it used the integration of COM+ 1.0 services and Microsoft Office charting controls.
- **Fitch and Mather 7.0** – The third version moved this solution onto .NET Framework technologies, concentrating on interoperability, deployment, and performance.

Note: The transactions, database, and stored procedures in Fitch & Mather 7.0 are essentially the same as in Fitch and Mather Stocks 2000.

For more information on Fitch and Mather 7.0, see the Web site at *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/f_and_m/html/vxoriArchitecturalOverview.asp*.

Key Technical Issues

The key technical issues in the Fitch and Mather sample application are:

- Security
- Distributed operation
- Reliability
- Scalability
- Performance
- Interoperability with legacy technologies

The Fitch and Mather Web site demonstrates the operation of a .NET Framework application without introducing excessive complexity. This makes it ideally suited as an example for use in this guide.

Note: There are elements of the Fitch and Mather sample application that are for demonstration purposes only and are not design guidelines. For example, the distributed design highlights .NET Remoting procedures. You would not implement .NET Remoting in that fashion in a live environment.

Technologies Used

The Fitch and Mather application uses the following technologies from Windows 2000, the .NET Framework and from legacy applications:

- ASP.NET
- Web Forms
- XML Web services
- Server Controls
- C# and C++
- COM+ 1.0 Services
- Internet Information Services
- ADO 2.7 and ADO.NET
- Enterprise Templates
- .NET Remoting
- SQL Server™ 2000

Architectural Diagram

The Fitch and Mather 7.0 example architecture divides into three tiers, which correspond to the recognized levels with the classic n-tier design. These tiers are:

- Presentation tier
- Business tier
- Data tier

Note: The tiers do not necessarily correspond to the physical location or number of computers deployed to support the application. Rather, they correspond to the logical design of the application.

Presentation Tier

The Presentation tier presents data to the user and optionally permits data entry and manipulation. The two main types of user interface for this layer are the traditional application and the Web-based application. Modern Web-based applications often contain most of the data-manipulation features that traditional applications use, through features such as dynamic HTML, client-side data sources, and data cursors. However, the implementation of Web-based applications results in more network traffic for the system because of the distribution of components among different computers.

In the FMStocks application, the Presentation tier uses Microsoft ASP.NET Web Forms and Server Controls and the Microsoft Visual C#™ development system. This combination presents the HTML-based user interface to the online customer.

Business Tier

The Business tier (also referred to as the business services layer) is where your COM and COM+ components reside. The components in this tier enforce the business and data rules that the application requires. All applications can use business components and you can move components to different physical or logical locations as response time and other rules require.

In the Fitch and Mather scenario, the Business tier hosts the business logic that handles accounts, buying and selling shares, and researching companies. The Business tier components use.NET assemblies built in Visual C#. You can distribute these assemblies across multiple computers using .NET Remoting.

The General Accounting Module (GAM) carries out account management functions implemented using the Microsoft Visual C++® development system. Using COM+ distributed transactions, the GAM can connect either to Microsoft SQL Server 2000 or Oracle 8.0 databases.

Data Tier

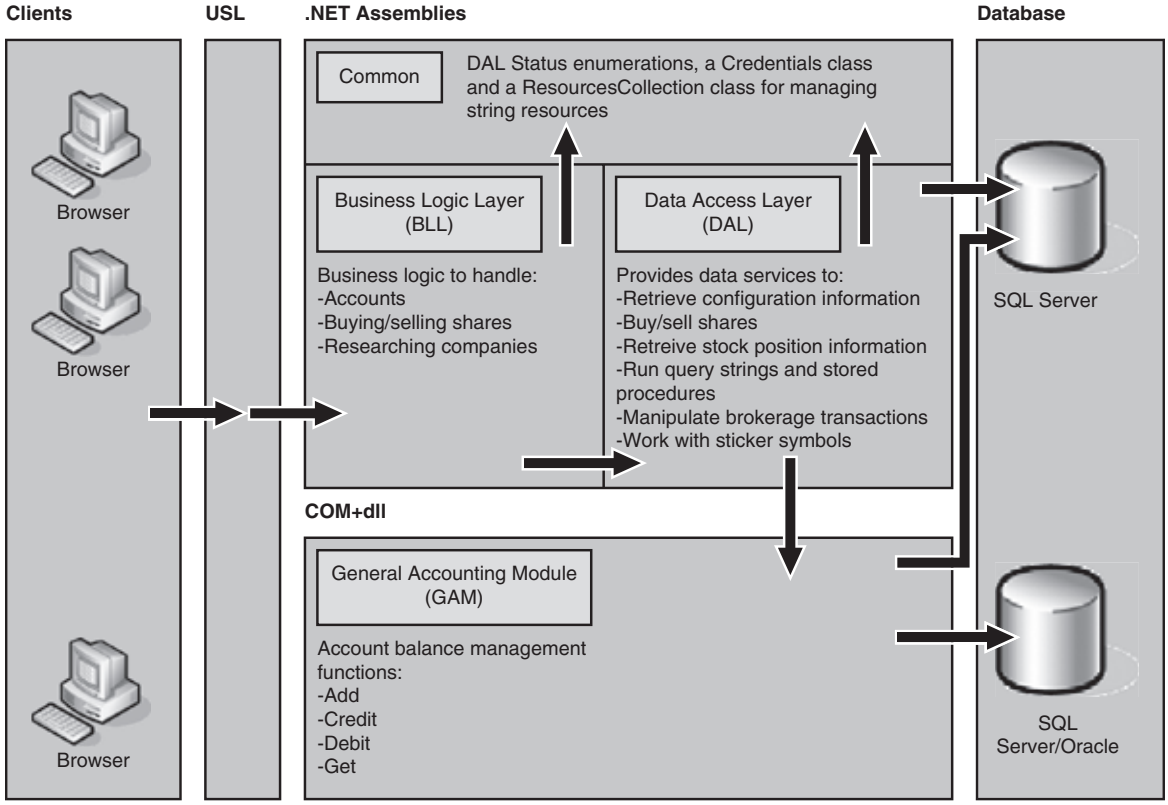
Sometimes known as the data services layer, this tier consists of the Database Management System (DBMS). Connections to the Data tier usually originate in the Business tier, but some Presentation tier applications can access the Data tier directly. This tier consists of data access components that provide resource sharing and allow user access without installing the DBMS libraries or Open Database Connectivity (ODBC) drivers on each client computer.

In FMStocks, the Data tier provides data for the Business tier, storing information resulting from user transactions. ADO.NET provides linkage to the data storage, with ADO 2.6 for connections to the GAM.

With the FMStocks application, the data storage itself can be on either SQL 2000 or on a combination of SQL 2000 and Oracle 8.0. A SQL Server 2000 database uses

stored procedures to record user transactions, with a SQL 2000 or Oracle 8.0 database storing accounting data.

Figure 1.2 summarizes this architectural information.



Note: Arrows to be interpreted as "... using ...". that is .:The Business Logic subsystem is using the Common subsystem

Figure 1.2
Fitch and Mather Architectural Diagram

Environments

In addition to the tiers, you can identify two environments that are not part of the classic n-tier model.

- Management environment
- Directory services environment

Management Environment

In the Fitch and Mather scenario, the management environment primarily consists of Microsoft Operations Manager. Other components of the management environment may include Application Center 2000 Server, the Virtual Private Networking (VPN) remote access server, and other third-party products that provide management services, such as Appmetrics for Transactions® from ExtremeSoft.

Directory Services Environment

The other main identifiable component is the directory services environment. The directory services environment provides user account management and authentication services to components in any other part of the infrastructure. Active Directory provides the directory services environment for the FMStocks application.

Network Diagram

Although you can install and run all the components of the FMStocks sample application on one computer, you would normally implement FMStocks as a full n-tier distributed deployment using .NET Remoting. Figure 1.3 shows a layout of how you might do this in a test environment.

Note: This design is not a template for a production system.

For more information on suitable network configurations for n-tier data centers, see the documentation for the Internet Data Center design at *http://www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/idc/default.asp.*

For more information on the deployment options for the Fitch and Mather sample application, see the documentation at

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/f_and_m/html/vxconfitchmathernetdeployment.asp.

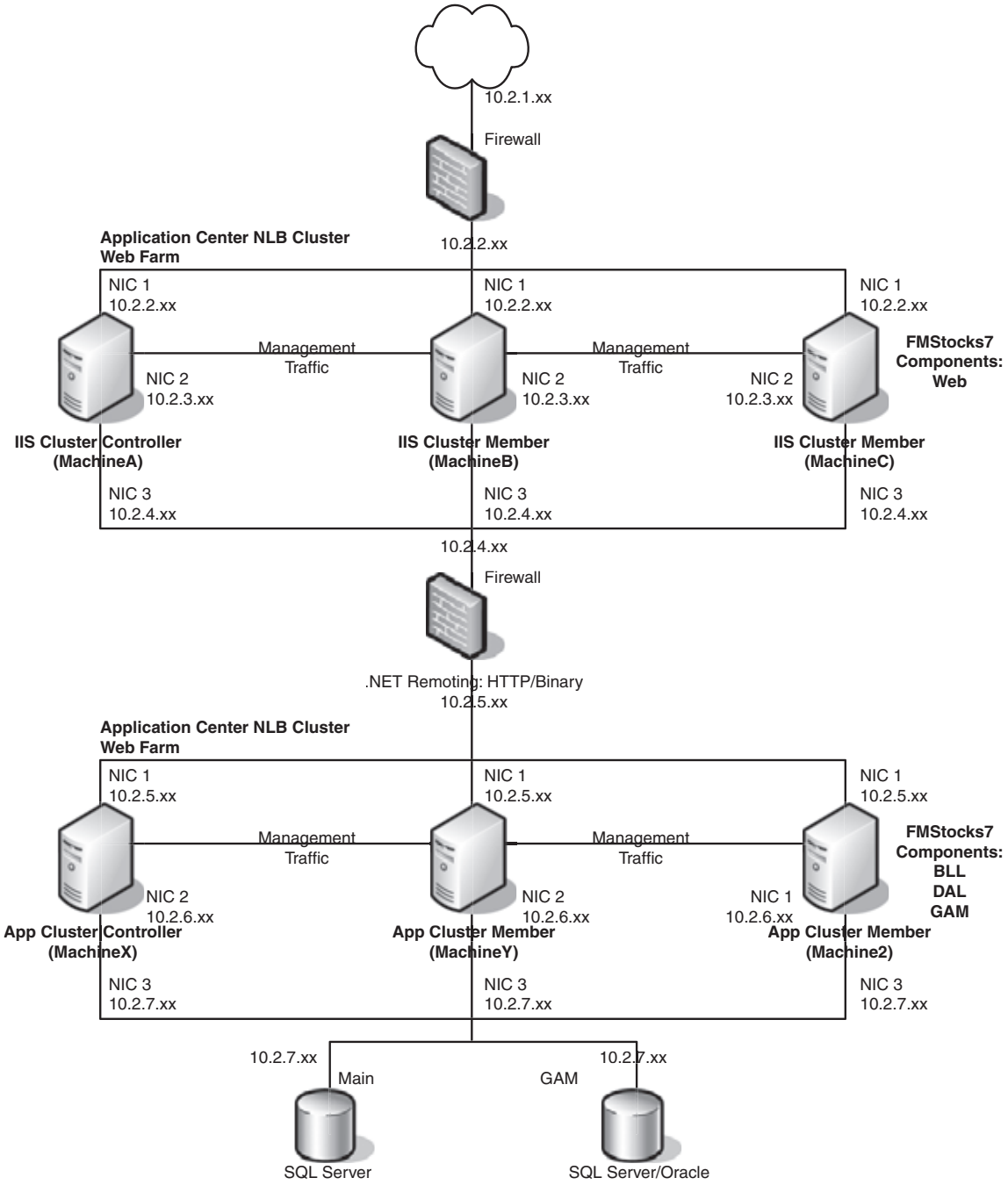


Figure 1.3
Fitch and Mather 7.0 Network Layout

End-User Activity

Because the Fitch and Mather application is aimed at the general (albeit stock-owning) public on the Internet, user actions are a main component of the application itself. Figure 1.4 shows the top-level activity diagram, representing a user's possible interaction with the Web site.

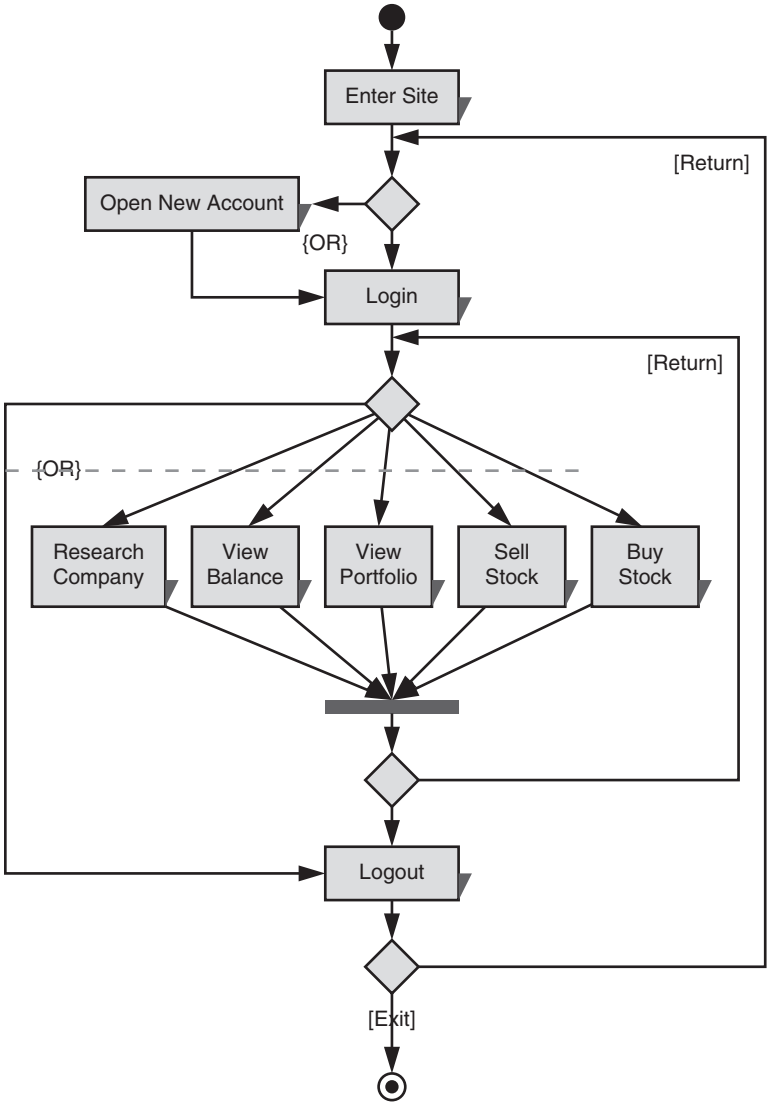


Figure 1.4
End User Interaction with FMStocks Application

Defining Usage and Test Scenarios

It is now time to look at the sort of users that make up the bulk of Fitch and Mather's clients and the profile of each user type's activity once logged onto the Web site. This information will then provide the data for the test scenarios in later chapters in this guide.

Defining User Profiles

Fitch and Mather has defined three main types of user that appear regularly their Web site. These are:

- The Browser
- The Buyer
- The Seller

Each of these user types has a particular usage profile and habits.

The Browser

The Browser seems to do just that—browse for information, and lots of it. In fact, it seems like the Browser is perpetually searching for that elusive extra bit of information and maybe, just maybe, they will eventually make a trade. One day.

The Buyer

The Buyer is confident. These stocks are definitely just about to go up. In fact, the Buyer is so confident that these stocks are a good investment that he or she will be tipping them heavily in their widely syndicated newspaper column on what are the hot stocks to buy this week. Buyers normally make their appearance at the end of the trading day once they have identified the good investments.

The Seller

The seller wants to get rid of their portfolio. And fast. In fact, they will sell anything, including their granny (if they haven't already listed her on eBay). Sellers normally come out of the woodwork first thing in the morning, get rid of lots of stocks, and then disappear. Quite often, buyers turn into sellers when they realize that the stock they have just bought is rubbish and try to offload it. Therefore you will see a surge of buying activity first thing in the morning and a smaller one just after the buying frenzy towards the end of the day.

Usage and Test Scenarios

When creating the monitoring test cases contained in this guide, the development team used the following three activity sequences as the main scenarios.

- Viewing account information
- Purchasing stock
- Selling stock

Appendix A shows the detailed steps for each test scenario.

Note: If you have installed the Fitch and Mather application, you can carry out the steps in each scenario by connecting to *http://yourwebserver/fmstocks7*.

Loading Fluctuations

A typical day at Fitch and Mather will produce the sorts of activity indicated in Table 1.2. This shows the approximate number of users of each type logged on at a particular time. Figure 1.5 shows this information in graphical format.

Table 1.2: User activity by user type over a typical trading day

Time (PST)	Browsers	Buyers	Sellers
00:00	200	20	25
01:00	100	10	20
02:00	80	8	18
03:00	50	5	15
04:00	30	3	10
05:00	45	5	5
06:00	50	5	5
07:00	150	15	10
08:00	550	55	30
09:00	650	65	180
10:00	730	73	70
11:00	670	67	40
12:00	590	59	20
13:00	650	65	30
14:00	700	68	50
15:00	800	75	65

Time (PST)	Browsers	Buyers	Sellers
16:00	750	200	70
17:00	600	60	75
18:00	550	55	90
19:00	500	50	85
20:00	480	48	75
21:00	350	35	60
22:00	300	30	50
23:00	200	20	35

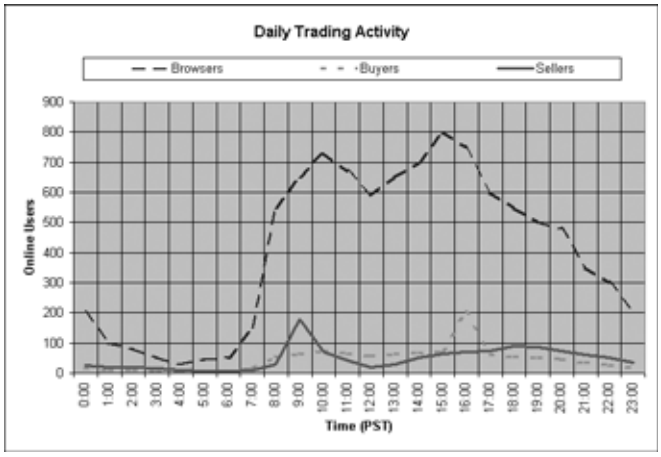


Figure 1.5
Typical Daily User Activity Chart

Summary

In this chapter, you covered the content of Operating .NET-Based Applications. You reviewed the tools and components that make up the .NET Framework and the environment in which you would deploy .NET-based applications. You looked at a sample organization that provides the scenario for of this book. Finally, you reviewed the business scenario and the example .NET Framework application that you will use throughout this material.

Appendix A – Test Scenarios

Viewing Account Information

When viewing account information, users log on, look through their portfolio, check out the financial news, and generally surf around the Web site for about 10 minutes before logging off.

Test Scenario 1 involves the following steps:

1. Connect to the Fitch and Mather web site.
2. Log in with the user's credentials
3. Click **View Summary**.
4. Click **View Portfolio** on the summary page.
5. Click **Home**.
6. Click **Financial News** and browse through this page.
7. Click **Home**.
8. Click **Managing your Money**. Browse through this page.
9. Click **50 Best Stocks in the World** and read this page from the MSN Web site.
10. Click **Logout**.

Purchasing Stock

In scenario 2, the main focus is on buying shares. The user does some browsing and checking on what they are going to buy before making their purchase. Again, for test reasons, the user decides to buy 10 shares in Coca-Cola® and 20 shares in Gillette.

Test Scenario 2 involves the following transactions:

1. Connect to the Fitch and Mather web site.
2. Log in with the user's credentials.
3. Click **Managing your money**.
4. Click **News** under Coca-Cola®.
5. Click **Buy Stock**.
6. Click the field marked **Ticker** and then type in "**KO**"
7. Press the Tab key to go to the **Shares** field and then enter **10**.
8. Click **Purchase**.
9. Click **Buy another stock**.
10. Repeat the procedure using the stock symbol **G** and enter **20** as the number of shares to buy.

11. Click **View your Portfolio** and check that the Coca-Cola and Gillette stocks are now part of the portfolio.
12. Click **Logout**.

Selling Stock

Selling stock is very like scenario 2 but here the user is in the market to sell. After a quick browse, they decide to divest themselves of their remaining Intel and Microsoft stocks before logging out.

Test scenario 3 consists of the following steps:

1. Connect to the Fitch and Mather web site.
2. Log in with the user's credentials.
3. Click **Sell Stock**.
4. Click **INTC** and then click the **Sell** prompt.
5. Type in **2** as the amount to sell then click the **Submit Order** button.
6. Click **Sell Another Stock**.
7. Repeat steps 4 to 5, using **MSFT** as the stock to sell and **4** as the number to sell.
8. Click **View your Portfolio** and confirm that the stocks have been sold.
9. Click **Logout**.

Module 1

Monitoring .NET-Based Applications

2

Application Monitoring Concepts

Introduction

In this module of *Operating .NET-Based Applications*, you look at how you can implement monitoring, alerting, and notification on Microsoft .NET-based applications. This ensures that you receive timely and relevant information on the status of software components running in your data center. You also look at programmatic techniques that produce monitoring information from within .NET-based applications. You see how to use the features of the Microsoft Windows 2000 operating system and other tools to ensure prompt and accurate notification and reporting of application monitoring issues. Finally, you review methods of ensuring that you and your staff can take prompt and efficient action once you receive a monitoring notification.

Effective notification goes hand-in-hand with monitoring because there is little point in having the most precise monitoring systems in the world if you have no idea when things aren't working the way they should. Notifications also need to be delivered promptly and through appropriate means (for example, an e-mail message telling you that your Exchange server is down might not get delivered very quickly).

You may need to publish regular bulletins on the health of your services, and this module looks at the most efficient ways of providing this information. Whether the final destination of the bulletin is an Intranet Web site or a flashing neon screen that announces the current availability level every second, effective reporting needs careful analysis of who receives the bulletins and the level of detail required.

Note: The prescriptive guidance in this module uses the Fitch and Mather scenario described in Chapter 1, "Introduction." This scenario is also the basis for the code samples using the Fitch & Mather Stocks (FMStocks) Web application.

In this chapter, you examine the concept of application health and look at why you need to monitor for application health. You then review monitoring terminology and methodology before covering general concepts related to monitoring. Finally, you look at application errors and the circumstances under which they arise.

Note: This module assumes that you are already implementing system-level monitoring targeted at the computers within your data-center environment, but that this monitoring does not specifically target applications. This system-level monitoring will be a component of your current Service Level Agreement (SLA).

Monitoring involves the following areas of IT:

- **Network monitoring—**Monitors Wide Area Network (WAN) and Local Area Network (LAN) devices such as routers, LAN switches, hubs, and firewalls.
- **Server Monitoring—**Monitors the servers' hardware. Most hardware vendors provide tools that monitor server hardware components and alert on failure conditions, such as power supply overheating or failure, hard-disk errors or failure, and so on.
- **Platform Monitoring—**Monitors the operating system. This area of monitoring is important because platform health directly affects application health.
- **Application Monitoring—**Monitors applications created in-house or purchased from external suppliers. You can partially monitor applications by proper platform monitoring. However, to monitor the application itself and gather information directly from it, you need to instrument the application.
- **Service Monitoring—**Monitors the user experience. This type of monitoring is complex because it touches on almost all areas in IT, such as the network, servers, platform, and application. This kind of monitoring usually tracks SLAs.

You can also monitor availability and performance. For example, network monitoring can focus on the availability of routers and LAN switches or on the performance of these devices.

This guide focuses on monitoring the health of applications that run on the .NET Framework, version 1.0 SP2. As you will learn, this guide uses platform monitoring to create the initial monitoring picture and then moves to application monitoring, by instrumenting the application. Instrumentation enhances overall application monitoring by providing detail that platform monitoring cannot provide.

Figure 2.1 illustrates the five monitoring areas in IT. This module focuses on the shaded areas.

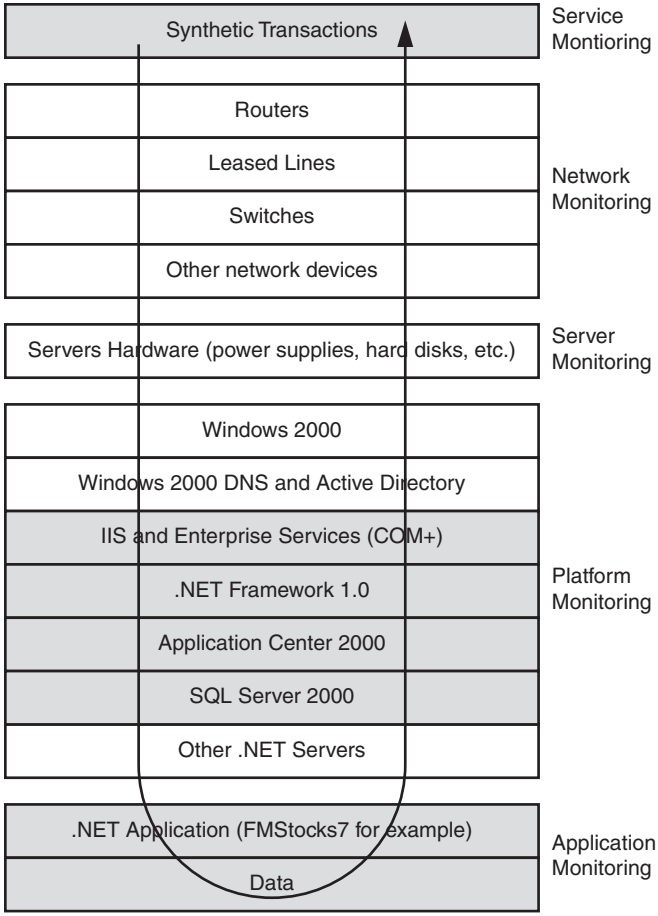


Figure 2.1
Shaded parts show the areas that this guide covers

Defining Application Health

Application health is an important concept in application monitoring and service reliability. But what is application health and how do you measure it? How do you separate application health issues from all of the other reliability issues that could affect your infrastructure?

At its simplest, an application is healthy when it provides the correct response to a user action or a request for resources within an acceptable period of time. The resource requester could be a user, in the case of a Presentation tier application, or another .NET-based application, such as a Business tier component or a Data tier database.

Characteristics of Healthy Applications

So is a healthy application one with a shiny coat and a wet nose? Or does it just need to have an all-over glow and eat plenty of organic vegetables?

A healthy application obeys all or most of the following rules:

- It is programmatically efficient (for example, no unnecessary looping and no needless resource-hungry API calls).
- It makes best use of the application environment.
- It makes the fewest demands on the infrastructure while still delivering the required performance.
- It makes resources available as soon as it has finished using them.
- It uses memory management techniques efficiently, such as using the pagefile when appropriate.
- It does not adversely affect other applications.

Characteristics of Unhealthy Applications

You can diagnose that an application is in an unhealthy state when it:

- Does not return a response.
- Returns the correct response, but too slowly or with inconsistent performance.
- Returns the incorrect response.
- Refuses to let go of resources.
- Makes inefficient use of operating system features, such as the virtual memory manager in Windows 2000.
- Affects the operation of other applications.

Out of the above examples, the most severe case is when an application does not return a response. This means that part of the application has crashed and cannot respond to the user or that the application is in a hung state. You should monitor for both situations.

If the application gives the right response but is unable to respond quickly or consistently, concurrent service requests are overwhelming the application or the application is programmatically inefficient. Alternatively, the computer itself is overloaded, affecting all applications running on that computer. In this situation, you need to diagnose further and check to see which of these factors applies. Monitoring the right set of metrics can alert you to the fact that users are experiencing this problem.

Sometimes applications do not release the resources they used when doing their work. An example of this is a memory leak, which can occur when repeated calculations cause an application to demand more and more physical or virtual memory, eventually causing the host computer to halt.

A component can affect other applications by tying up the processor or other resources, such as rows in a database table. The effect is that the entire system runs slowly or, even worse, all subsequent user requests hang until the blockage clears.

Security issues can also affect application health. For example, distributed denial of service attacks can prevent an application from responding to genuine requests. Even applying security can affect application execution and application health. For more information on securing .NET-based applications, see Module 2 of this book.

Several factors can compromise application health. Using the right set of monitoring metrics and techniques enables you to distinguish between these different scenarios and take appropriate action.

Monitoring Application Health

Techniques that monitor the health of an application fall into the following two categories:

- Gathering information from the systems outside of the application
- Adding code to your application to provide information about the performance of the application

Chapter 3, "Selecting Data Sources," looks at the first category and shows you the most significant counters and events provided by the operating system. Chapter 4, "Instrumenting .NET-Based Applications," looks at the second category and describes how you can extract application-specific detailed information.

In addition, you should monitor your application starting from the point at which the user first interacts with the system through to completion. This is end-to-end monitoring uses *synthetic transactions*, which check for successful completion of specific actions. Synthetic transactions mimic real-life user interactions with a service, thus providing excellent warning on the failure of any component within the infrastructure.

Why Monitor Application Health?

Application health is extremely important because an unhealthy application can affect much more than the application itself. Organizations increasingly rely on their computing infrastructures, and you cannot afford to have vital components that are not performing properly. The operations staff incurs costs from constantly reacting to problems, but the organization pays a price as well.

If an application is unavailable or slow, the company pays for the lost productivity of the knowledge workers who use internal applications. If the application generates revenue, unavailability or poor performance means lost income and reduced customer confidence.

Application health monitoring lets you know if your application is running correctly and efficiently. By knowing this, you can increase the overall availability of your application.

The following are some of the reasons you should monitor your .NET-based application:

- Monitoring determines whether you are achieving the levels of availability and performance that are specified in your SLA.
- Application downtime is expensive, and it could affect a company's profit, reputation, and competitiveness.
- Up to 80 percent of the total cost of solving a problem is spent identifying the cause of the problem.
- Monitoring application health is critical to identifying the root cause of a problem.
- Monitoring application health increases the availability and reliability of your .NET-based application by helping to find these root cause failures.
- Many problems only manifest themselves in the production environment.
- You'll know of a service outage before your users do.

Monitoring application health also helps you understand application performance and can help you plan for increased application usage, also called capacity planning. For more information, see Module 3, "Sizing and Capacity Planning for .NET-Based Applications."

Whatever your ultimate monitoring goals are, you should implement a health-monitoring solution first and then augment or extend your health monitoring to cover performance and capacity planning afterwards. These issues become irrelevant if your applications are too unhealthy to function correctly.

Defining Monitoring Terminology

The following terminology is used in this module:

- Tiers
- Logical divisions
- Coarse-grained and fine-grained monitoring

Tiers

Tiers are physical sections within the architecture. In the Fitch and Mather scenario, there are three tiers:

- Presentation tier
- Business tier
- Data tier

These correspond to the three standard tiers in the *n*-tier distributed architectural model used in the Microsoft Internet Data Center (IDC) framework.

Note: Although this book refers to a physically distributed design where the tiers are separate from each other, it is possible to implement .NET-based applications in a three-tier framework on any number of computers, even as few as one. The Microsoft Visual Studio .NET documentation explains how to do this.

Logical Divisions

Logical divisions are virtual sections within the application architecture. In the context of the Fitch and Mather scenario, you can identify six distinct logical divisions:

- Internet Information Services (IIS)
- ASP.NET
- Managed code components
- Serviced components (managed and unmanaged code)
- .NET Remoting
- Data

Each logical division will have its own monitoring counters and requirements. Figure 2.2 on the next page shows the relationship between tiers and logical divisions.

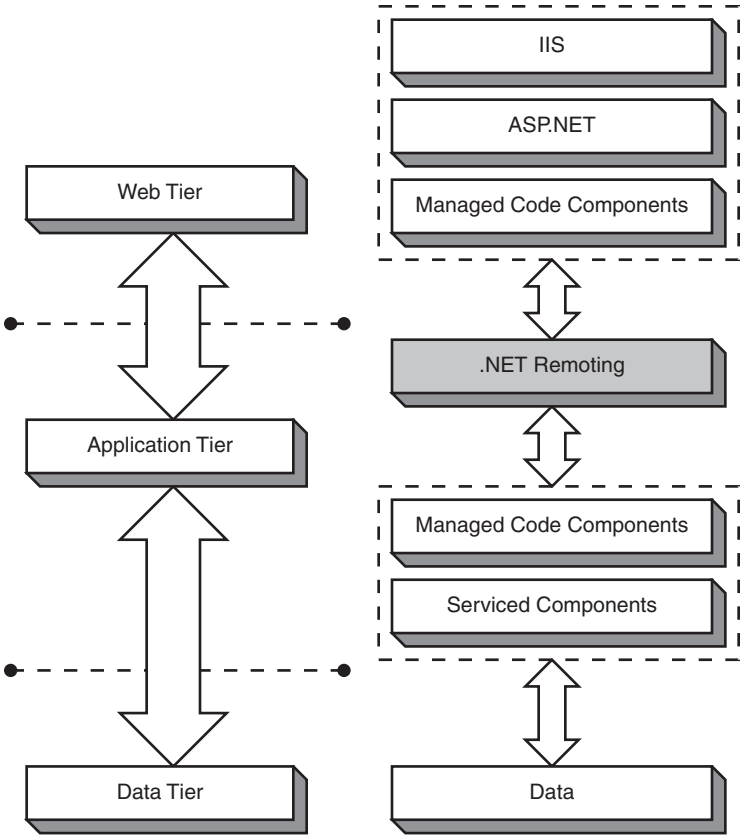


Figure 2.2
Relationship between tiers and logical divisions

Internet Information Services

IIS (the Web server) responds to HTTP requests; it replies to an incoming client request, and then invokes the specific ASP.NET pages. ASP.NET then executes the code on the requested pages and generates the proper HTML content, which returns to the client through IIS. Monitoring at the IIS logical division tends to focus on counters that reflect the user's experience of the front-end Web interface.

ASP.NET

ASP.NET is the heart of the application on the Web tier. For standard .NET-based applications, ASP.NET takes the raw HTTP requests from IIS, executes the code on the pages that make up the application, and returns the dynamic pages back to IIS. Monitoring ASP.NET tends to be more complex than monitoring other applications because you may need to look at the composition of the pages, as well as the speed at which pages are returned.

For Web services, ASP.NET handles the incoming SOAP request. From a monitoring perspective, the difference between a Web service and a standard .NET-based application is minimal.

Managed Code Components

Managed code components execute within the framework of the .NET common language runtime. (For more information on the common language runtime, see Chapter 1, "Introduction.")

Monitoring the .NET common language runtime provides the majority of monitoring information for managed code components. For more detailed information on managed code components, use application instrumentation. Chapter 4, "Instrumenting .NET-Based Applications," shows how you can add monitoring code to applications.

Serviced Components

In the .NET Framework, you can write serviced components using managed or unmanaged code. Many of the applications currently running in data centers use unmanaged code serviced components, although you probably know them as COM+ applications. Managed code serviced components are created using the .NET Framework and use all of the same services as traditional COM+ components. Both types run under Component Services (COM+).

Monitoring individual components allows you to break down the work being done by your application into the types of transactions running inside it. This lets you better understand the workload distribution both between applications and within a single application. For example, you can discover which applications are most active during which periods of the day.

.NET Remoting

.NET Remoting is one of three techniques that you can use to link the tiers in your infrastructure together. You can also use Distributed Component Object Model (DCOM) services, or make requests to a .NET Web Service in another tier.

In the Fitch and Mather scenario, .NET Remoting is the chosen method for providing inter-tier connectivity because of convenience and security. You can configure .NET Remoting to use HTTP over TCP, which makes it easier for you to configure internal firewalls separating the Web and application tiers.

Data Logical Division

The data logical division contains your database management system (DBMS) and your data. Here you are looking at counters that, in the case of the Fitch and Mather scenario, target SQL Server. Alternatively, you might be running another database environment with its own monitoring framework.

Remember that the data logical division often greatly affects overall performance. Therefore, monitoring at this level needs to include not only the DBMS but the computers on which the DBMS runs.

Coarse-Grained and Fine-Grained Monitoring

Coarse-grained monitoring shows high-level, general data (typically concerned with computer operation). *Fine-grained* monitoring shows the low-level, detailed view of what is happening to an individual application. However, coarse-grained and fine-grained monitoring often look at similar areas, and they normally use the same system-provided counters, but view them from different perspectives.

Consequently, coarse-grained monitoring information focuses on the computer system as a whole, looking at counters such as total processor usage, free disk space, physical memory usage, and so on. Fine-grained monitoring typically covers areas such as processor usage by process or thread, disk space by database, and the working memory set for an individual .NET-based application.

Chapter 3, "Selecting Data Sources," shows the counters that you can use to build either a coarse-grained or a fine-grained view of your infrastructure. Which level you use at any particular point is a matter of judgment. Application design best practices recommend that you design the monitoring framework when you develop the application. However, if this approach is not feasible or not followed, implement coarse-grained monitoring initially, reassess the situation, and then add further fine-grained monitoring as appropriate.

Note: With monitoring, as with so many other areas in life, you will probably find that the Pareto 80-20 rule applies and that 80 percent of your application problems arise from 20 percent of the possible causes. Therefore, you should concentrate your monitoring effort towards those 20 percent of causes, once, of course, you have identified which causes they are.

Application Monitoring Methodology

Before you start monitoring your .NET-based applications, define a methodology for application monitoring. Concentrate on the important areas that directly affect the operation of the applications in your data center and ignore unimportant or inappropriate monitoring sources.

The methodology you use for monitoring your .NET-based application is quite simple:

1. Establish the monitoring focus.
2. Determine the physical and application architectures to clarify where the required services and components are running.

3. Identify what sets of information provided by the Windows platform and middleware technologies you can use (for example, the .NET Framework and COM+).
4. Monitor this system-provided information to create a baseline of the application's normal range of operations.
5. Augment this picture using synthetic transactions or application-specific instrumentation for critical or complex areas particular to the .NET-based application.

Establishing the Monitoring Focus

You must determine the reasons you need to monitor your applications. It could be for one or more of the following:

- General health
- Performance
- Capacity planning
- Service availability

When monitoring, have clear goals. This keeps you from either collecting information that is unnecessary or from missing a key piece of information.

Determining the Physical and Application Architectures

Ideally, you should understand your current physical architecture in terms of the hardware components of each tier and the connections between the tiers. You also need to understand the application architecture, and know how the applications running on each tier interact with other components, either on the same tier or on different tiers.

Identifying System Information

You need to collect information already provided by the various technologies that your application uses. For example, you might already be collecting monitoring information to assess the general health of Windows 2000 and your network. Augment this system information with additional data describing specific technologies on which your application relies. For example, middleware technologies that your application uses, such as the .NET Framework or Enterprise Services, provide monitoring information that you can collect.

The advantage of starting with system-provided data is that it works for any application. Because many applications do not provide their own instrumentation, this may be the only possible way for you to monitor what is happening.

Building a Baseline

Next, you must build a baseline of the application's normal operations. This baseline will consist of measurement data and the occurrence (or more often, the non-occurrence) of particular events. It is vital that you understand what normal operation is so that you can identify what is abnormal.

Augmenting the Picture

After you've created a baseline picture using system-provided data, you can then use additional techniques to gain a more detailed understanding of your application. These additional techniques include:

- Artificial simulation and stimulus
- Instrumenting applications

Using Artificial Simulation and Stimulus

Traditionally, one thinks about monitoring in the more literal sense of measuring system variables, rather like a voltmeter or oscilloscope on an electrical circuit. But monitoring in this traditional, literal sense can only provide evidence of what is happening or has happened.

You can use a class of monitoring that measures the presence or absence of the application as a whole. For this type of approach, you need to employ techniques that artificially stimulate the application. For example, you could monitor the number of pages per second that the Web service processes. If the number of pages per second goes to zero, is that because no one is using the Web service or because some part of the connection upstream of the Web service is down?

An artificial stimulus application can answer this question for you because it provides a known load at a known time. If you do not see that occur, you can conclude that there is a problem preventing access to your .NET-based application.

Instrumenting Applications

System-provided information does not always give a complete picture of your application's performance. There are often certain pieces of information missing or problem areas that require more detailed, application-specific information.

You can obtain this information by initially designing or modifying the application itself so that it includes the necessary instrumentation code to generate specific monitoring information. Chapter 4, "Instrumenting .NET-Based Applications," describes the tools available in the .NET Framework that add this functionality.

Multiple End-to-End Monitoring Techniques

Regardless of the monitoring approach you take, it is recommended that you have numerous views of your application's health available to you. One tool could miss a critical symptom that other tools might catch. However, you need to be aware of what the appropriate level of health monitoring is.

Use all tools at your disposal in concert to determine the health of your application. Even if your application seems completely healthy today, increased user load tomorrow can uncover new problems. The rise in CPU utilization and disk loading may cause an application to perform improperly, or drastically reduce the performance of other applications running on the same computer.

The continued monitoring of this data is the key to preparedness. Be continually on the watch for symptoms of a problem. If you catch the symptoms early enough, you may be able to prevent a serious problem.

Monitoring Issues

When you monitor an application, you have to trade application throughput against the amount of monitoring data you collect. The very act of collecting monitoring data consumes system resources. The more monitoring data you collect, the more system resources are needed, which reduces the resources available to the application.

In addition, there are many instances where the monitoring process affects the resource that you are monitoring (for example, when you monitor processor usage with Windows 2000 System Monitor). If you monitor a computer from its own console, collecting, interpreting and, most importantly, displaying a changing performance-data graph increases the load on the computer's processor, which distorts the CPU usage results being displayed. Therefore, in most environments you should monitor computers remotely.

Remote monitoring is not a universal solution, however. If you are monitoring network activity, the packets passed back to the monitoring workstation increase the number of packets recorded on the monitored computer. In this case, it is probably more appropriate to monitor locally. Analyze the type of monitoring that you are performing and decide whether data collection should be local or remote.

Monitoring Load

You need to decide on an appropriate monitoring load. The rule of thumb is that monitoring should take less than 10 percent of a computer's total CPU usage. For applications that need only a few simple metrics and a heartbeat check, this rule holds reasonably true. But with applications that represent more critical business functions, the question becomes "What information do I need?" or, more aptly "What information can't I afford not to know?" If you monitor your company's

mission critical application using less that 10 percent of the computer's total CPU, but you fail to detect that crucial event when the application hangs, then your monitoring is failing in its primary purpose.

Monitoring and management should not be seen as overhead, but as essential components in the efficient functioning of your .NET-based applications and services. Determine the appropriate level of monitoring load and then scale your hardware to suit this additional workload.

Generating alerts can affect the performance of the application, which is another concern when monitoring applications. This can generate additional events, which further degrades performance. Consider using monitoring methods that do not escalate the monitoring load.

As your experience with a particular .NET application grows, you may find operational areas where you need more detail. You can accomplish this by increasing the amount of monitoring in that area. You might choose a particular area because you've identified it as being problematic, or a critical area of an application that would cause major problems if it did not work.

Monitoring the Common Language Runtime

Monitoring the operation of the common language runtime is unique to .NET-based applications. Windows DNA-based applications written in the Microsoft Visual Basic or Microsoft Visual C++ development systems had runtime libraries, but those libraries did not provide extensive information about the runtime execution. The common language runtime provides information about many aspects of its execution, such as memory, networking, .NET Remoting, and locks and threads.

Another aspect of the operation of the common language runtime that is different from Windows DNA-based applications is that the common language runtime has built-in memory garbage collection. You may observe different memory usage patterns for your .NET-based applications than for your Windows DNA-based applications. For example, your .NET-based application's Private Bytes value may appear to grow and stay high like a traditional memory leak, but this could also be caused by the fact that that the garbage collector has not run.

However, the garbage collector cannot entirely get rid of all memory leaks. First, there may be legacy code running in the .NET-based application that has memory leaks. Second, there are coding practices, which if not followed, will cause a memory leak that even the garbage collector cannot reclaim.

Events and Metrics

The data you collect, whether it comes from a system-provided source or from application instrumentation, falls into one of two categories: events or metrics. Metrics are numbers that represent measurement of a variable, and that measurement has units associated with it. System Monitor counters show metrics: for example, the amount of free memory on the computer (measured in kilobytes or megabytes) or CPU utilization (expressed as a percentage).

Defining an Event

An event represents that something has occurred. For example, if a computer that runs Windows 2000 starts, the Windows 2000 Event Service writes a 6009 event into the System Event Log that records this occurrence.

The event has one or more pieces of data associated with it that describes the event, covering the what, when, and where. For example, the event recorded when a computer starts includes the name of the computer, the operating system, the current service pack level, and the date and time of the event.

Event Timing

There is a time element associated with both events and metrics. Time is important because it provides the context to evaluate the raw data. For example, when did the CPU utilization reach 80 percent or when did the system restart?

Time is also an important axis on which to draw a correlation between events and metrics. When you collect events, it is important to record the time that the event occurred, not just the time that you receive the alert. When you add application instrumentation, you must include a precise timestamp so that you can discriminate between two events that occur in quick succession.

Note: Accurate timestamps are essential for cross-correlating events, particularly in a distributed environment. Ensure that all computers are synchronized with an external time source.

Application Errors

Application errors come in a number of different forms, and errors can occur even in applications that have been thoroughly tested. During development, it is impossible for a developer to predict every possible error and then write code accordingly. Eventually every application, no matter how mission critical or well developed, incurs errors.

Note: Developers must use consistent error trapping and assign appropriate severity levels during application development. This greatly increases reliability and assists in the swift resolution of failures.

Errors can occur because of something as simple as a change in traffic volumes or usage patterns. Errors can also be produced when network component or server hardware fails.

Changes to the application environment can be a huge source of errors. These errors could be caused by rolling system changes into production, applying service packs, or upgrading operating systems. Even seemingly simple changes to a system component, such as installing a security patch, can seriously impact running applications. To reduce the risk associated with errors of this nature, use effective version control, and pilot any changes to the production environment.

Other error sources include:

- Amendments to database schemas
- Alterations in security levels that prevent COM+ applications from accessing resources
- Undetected bugs in production applications

Memory Leaks

Memory leaks are the most insidious form of application error. They rarely cause catastrophic failure, but slowly build up over time, reducing the effectiveness of all applications running on a computer. Memory leaks are normally caused by software that fails to return memory that it has requested.

This is particularly problematic where the function requesting the extra memory allocation is part of a programmatic loop, because the application requests more and more memory until the physical RAM or the pagefile (or both) is exhausted.

A simple system counter can easily monitor all of your applications for memory leaks. However, finding which application is the culprit and, most importantly, identifying the processes that spawn the memory leak, can take significantly more research. Chapter 3, "Selecting Data Sources," shows you how to detect issues such as memory leaks.

Common Language Runtime Errors

Common language runtime errors can occur in any application that uses the .NET Framework. The .NET Framework management pack lets you monitor common language runtime errors in any tier of your application. This book, however, focuses on those errors that tend to crop up in the ASP.NET logical division or in the .NET Remoting component.

ASP.NET allows your developers to create Web applications with any language following the Common Language Specification (CLS). Languages that follow the CLS can run on the common language runtime. The .NET common language runtime provides features such as garbage collection, secure code execution, structured exception handling, and more.

Your developers can take advantage of this language-independent, structured exception handling when writing ASP.NET applications. Structured exception handling uses try/catch/finally blocks, and forces you to deal with exceptions when they occur. If an exception crops up that you do not explicitly handle, a general ASP.NET exception occurs, which forces the application to terminate. The code cannot continue executing, creating an error page.

.NET Remoting enables common language runtime components in different application domains to talk to each other across a distributed environment. Errors with .NET Remoting and the common language runtime can be as simple as forgetting to add the reference to the runtime remoting dynamic-link library in the build project. Failing to do this will result in an error stating that the TCP remoting channels namespace does not exist.

Application Hanging

When an application hangs or fails to respond, it instantly affects any users that are interacting with that program. The hang might be caused by an application error, or it could be because part of the program is waiting for a response from another component or from the system. Alternatively, applications can hang if they get into continuous looping or as a result of an unhandled exception.

System counters can often detect an application that fails to respond because the operating system determines that a program has stopped executing. You can use several other techniques to detect whether applications are still operating correctly, including sending the application an instruction to see if the application processes your request.

Communication Errors

A variety of events can cause communication errors, from the most basic, such as a failed network card, to more serious problems, such as mismatched protocols, authentication issues, incorrect firewall configurations, or improperly specified sockets to non-functioning name-resolution services. Monitoring for communication errors can start from simple PING queries, move through simulating user interaction by creating and checking synthetic transactions, and end with instrumenting applications to report on communication health.

Summary

In this chapter, you reviewed the areas that you will be covering in the remainder of this module. You looked at the concept and importance of application health. You looked at the physical and logical divisions within the infrastructure and you understand the concepts of coarse and fine-grained monitoring. Finally, you covered monitoring issues that are specific to the .NET Framework.

3

Selecting Data Sources

Introduction

Effective application monitoring provides the correct level of useful information with an appropriate amount of detail in a timely manner without excessively impeding application performance. However, because monitoring places additional loading on the monitored object, you need to balance the level of monitoring against performance.

In this chapter, you learn how to collect data from existing sources. You cover how tiers and logical divisions can assist you in focusing monitoring effort and look at specific counters and metrics. Finally, you see how to use synthetic transactions to prove whether an application is functioning correctly.

Note: Having read Chapter 1, "Introduction," you know that there are minor differences between monitoring applications built on the Microsoft .NET Framework (.NET-based applications) and monitoring distributed applications built on the .NET Framework (.NET-connected applications). For example, synthetic transactions play a minor role in monitoring .NET-based applications but are critical when monitoring .NET-connected applications.

Throughout this chapter, you look at monitoring using the concepts of tiers and logical divisions within the application architecture. The examples directly relate to the Fitch and Mather scenario and the FMStocks application illustrated in Chapter 1, "Introduction."

Monitoring with System-Provided Data

Monitoring with system-provided data involves using the basic counters and metrics provided by the operating system and by components such as Microsoft Windows Management Instrumentation (WMI). Although this system-provided data might not give you the monitoring detail that you require, it is an important starting point for a number of reasons. For one, you are likely to monitor system-provided data as part of your data center operational practice. Additionally, monitoring system-provided data is generally simpler to implement and impedes application performance less.

Reviewing Current Monitoring Provision

When you identify metrics and events that you want to monitor in your .NET-based application, you are almost certainly collecting some of that data already. Hence, you might let your existing monitoring framework cover those particular metrics and events or you might include them when monitoring your application.

You should decide whether to duplicate the monitoring of certain metrics and events based on how you want to view your alerts and on what kind of reports you want to generate. For example, you might want to compare the number of ASP.NET request processes to the total system CPU usage. If you are using Microsoft Operations Manager (MOM) to gather the CPU information, you can easily combine the CPU information already being collected with the ASP.NET metrics. If you are gathering the CPU information using a different tool, you might duplicate the CPU metrics for completeness.

Working in Stages

When creating a monitoring picture of your application using system-provided data, work in stages. Start with a simple monitoring implementation that does not require extensive planning or a large amount of system resources. You can then monitor in greater breadth and depth over time rather than immediately plunge into a complex monitoring framework. Increasing monitoring levels over time makes creating an effective monitoring environment more likely and reduces the risk of implementing an overly complicated and unmanageable setup.

Your initial monitoring picture might be somewhat rough, but it should provide a base on which to build. As you become more familiar with the idiosyncrasies of your .NET-based applications, you can add further monitoring to create a more fine-grained picture, either for the entire application or for parts of it.

Monitoring Tiers and Logical Divisions

Simply viewing the health of your application from the perspective of the health of the underlying computer systems is inadequate. There are many circumstances in which the underlying computer might be trotting along quite happily while your application is lying on its back with its legs in the air. To obtain more information, look at the application architecture and monitor the tiers and logical divisions within that environment.

The application has a structure that allows viewing of the application's health. As you saw in the introduction to this book, the FMStocks application deploys onto three physical tiers, with the computers in each tier making up a cluster. Within each tier, you can further segment the computers into application logical divisions, as Figure 3.1 shows.

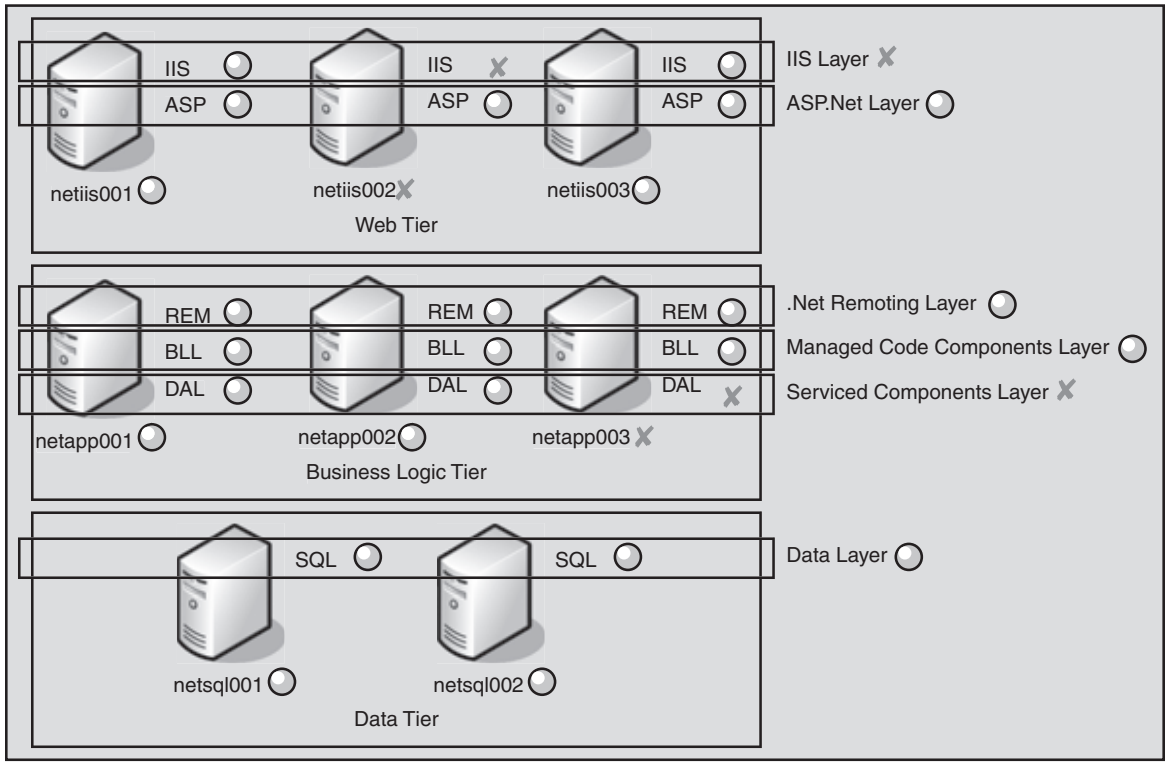


Figure 3.1
FMStocks tiers and logical divisions

Clustering multiple computers necessitates viewing the computers in the cluster both as a single entity and an individual computer. By monitoring layer-specific information, you can better understand the problems within your infrastructure. You can then aim analysis and corrective action to the elements of each layer.

Defining Monitoring Levels

When monitoring system-provided data, you can define two major levels of monitoring detail:

- Coarse-grained monitoring
- Fine-grained monitoring

It is important that you understand and appreciate the scope and effects of each type and when to employ it. This ensures that you use the correct monitoring level depending on the circumstances.

Coarse-Grained Health Monitoring

The goal of coarse-grained health monitoring is to ensure the functioning of each of the logical divisions (not tiers) within your application architecture. To do this, you need to take a layered, hierarchical approach.

The metrics you choose for this type of health monitoring are readily available and easy to collect. The objects and counters themselves do not provide much detailed information, but they also do not create significant overhead.

The simplest example of coarse-grained monitoring is checking whether the computer is running at all. In a TCP/IP environment, you can achieve this by using the PING protocol to check for the computer's IP address.

A more complex coarse-grained example might cover where the FMStocks application installs onto a number of computers forming a load-balanced cluster. In this situation, a coarse-grained monitoring goal might be to check whether the cluster is working, which you can do using Application Center 2000. To monitor whether the cluster is working at maximum efficiency, however, you must monitor each computer within the cluster. Doing so requires further inspection to get the level of detail that you want to create a complete picture of an application.

Note: Clusters are a special case for monitoring, as they need to be treated as both virtual and physical units. For example, in clustered implementations of Microsoft SQL Server™, you monitor the virtual server metrics as well as the metrics for each computer node.

Figure 3.2 illustrates this second view. In this diagram, a check mark (v) shows that the appropriate metric on the system is within acceptable values, and the *X* indicates that there is a problem.

Looking at the difference between Figure 3.1 and 3.2, you can see that the Internet Information Services (IIS) layer on NETIIS002 can account for the excessive CPU utilization. Hence, the IIS logical division does not function properly because of the NETIIS002 machine.

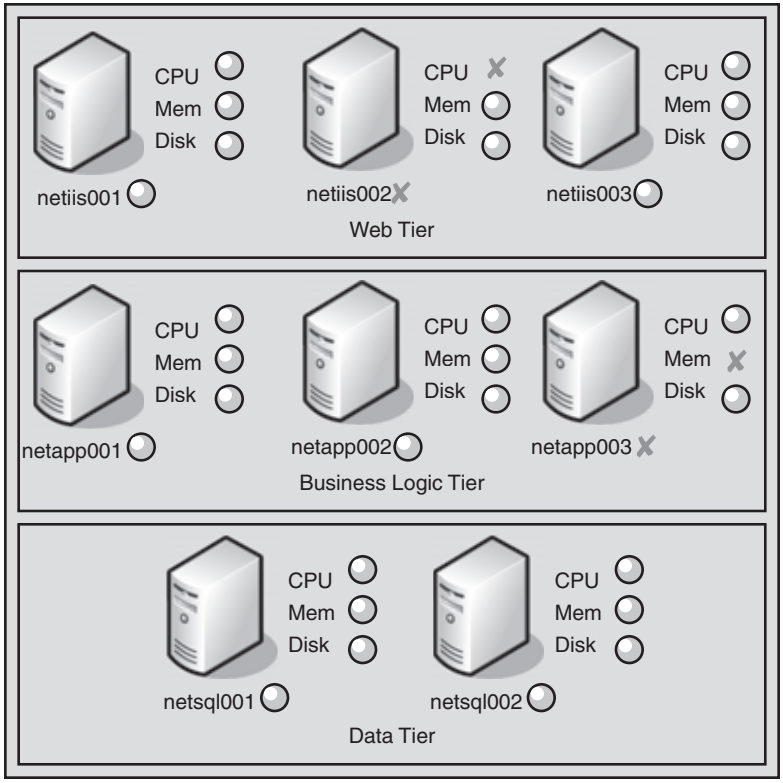


Figure 3.2
Coarse monitoring view of FMStocks showing acceptable and unacceptable components

Note: IIS not functioning properly on one node is different from IIS not functioning for the array. In the latter case, the load-balancing software compensates for IIS not running on one computer. Hence, you need to implement both node-level and cluster-level monitoring to get the complete picture.

The coarse-grained picture provides you with basic coverage of each of the major subsystems of your .NET application. In particular, the coarse-grained picture aggregates data to simplify your view of the systems. This is a good start, but you are likely to need greater levels of monitoring detail, particularly when applications start to produce errors.

Fine-Grained Health Monitoring

Fine-grained health monitoring helps separate the data first by application and then by transaction within each application. Creating this type of fine-grained picture allows your operations staff to see the relationship between applications. This approach addresses such issues as whether one application is demanding more resources than the others or whether the memory usage of one application is growing faster than the rest.

Coarse-grained monitoring can create only the overall picture and report that memory is being consumed somewhere; fine-grained monitoring can give you the information that you need to make detailed judgments about individual applications. The next topic shows one way to implement fine-grained monitoring.

Distinguishing Between Applications

In an enterprise computing environment, FMStocks would most likely be only one of many applications. Hence, one of the ways to add more fine-grained information is to separate the different applications running on each computer or cluster.

To explore fine-grained monitoring, assume that Fitch and Mather is part of a financial services group. Rather than just one .NET-based application, the organization runs brokerage (FMStocks) and commercial banking (FMBank) processes on the same computers within their data center.

Coarse-grained monitoring records excessive CPU usage in the IIS component and memory issues in the Data tier; however, coarse-grained monitoring fails to identify which application is responsible. To do so, the Fitch and Mather operations manager wants to collect health statistics separately for each application.

The FMBank and FMStocks applications share the same tier structure and layers but differ in the specific ASP.NET pages, components, and database tables that they use. You can visualize the separation of multiple applications as vertical divisions of the application layers.

To identify the CPU and memory problems in its current architecture, the Fitch and Mather operations staff implements the following changes to their monitoring framework:

- In the Presentation tier, they create the fine-grained picture by collecting metrics about the individual IIS Web sites and ASP.NET applications rather than the totals.
- In the Application tier, they group the components for each application and collect each group separately.
- Finally, they do the same in the Data tier by collecting statistics for each database separately.

Figure 3.3 shows an example of fine-grained monitoring.

When reporting on this information, the Fitch and Mather operations staff collates the FMStocks information for each tier. They then gather the same information for FMBank.

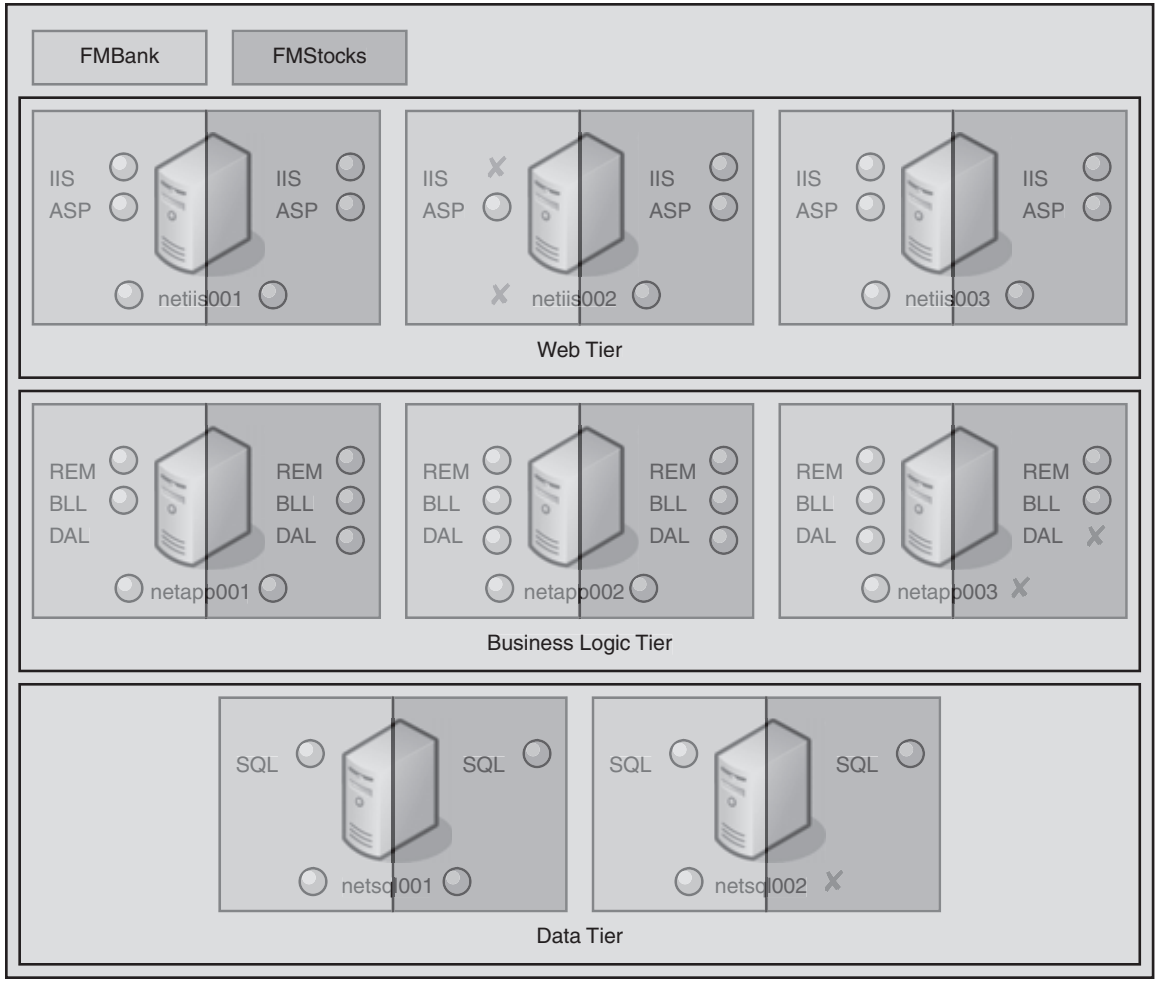


Figure 3.3
Multiple applications in the Fitch and Mather data center

As you can see in Figure 3.3, by creating a fine-grained monitoring picture and breaking down the health information by application, the operations staff diagnoses that the CPU problem on the computer NETIIS002 comes from the Web site for the FMBank application.

To round off your fine-grained health monitoring picture, you want to know which transactions within an application are succeeding or failing. By separating the transactions of FMStocks in the previous example, you can see that the problem with excessive memory use is in the component in the Data Access Layer (DAL) logical division, as Figure 3.4 on the next page shows.

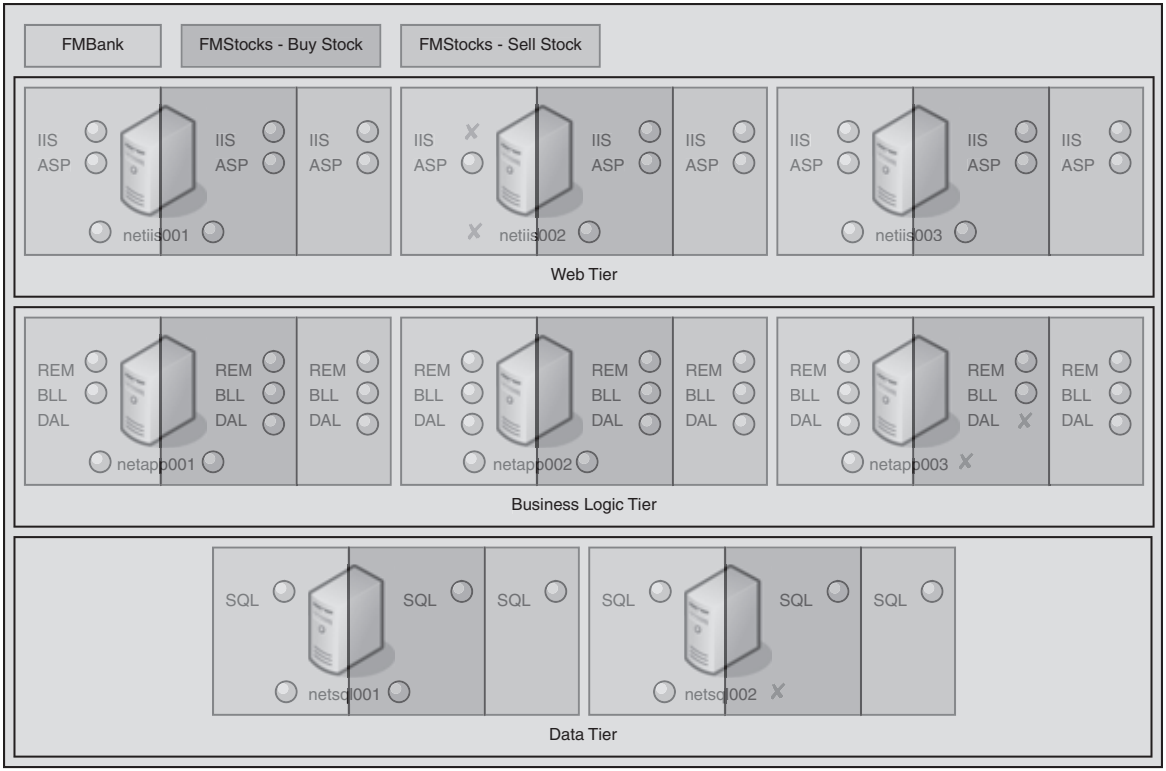


Figure 3.4
FMStocks application monitored by an individual transaction

By using both coarse-grained and fine-grained monitoring, you can identify and solve the application issues within your infrastructure.

There are tradeoffs, however, between the amount of health monitoring data you want to collect and the amount of resources it takes to collect the information. This is no different for .NET-based applications than for the systems and applications that you are monitoring today. Sadly, you will probably find that when all is going well, you have too much information, and when there's trouble, you never have enough. You have to balance the monitoring levels that you configure and the overhead generated by that monitoring.

Monitoring Logical Divisions

If you record a simple set of counters for a computer, such as CPU or Memory loading, you might find that none of these counters report heavy usage. Without further information, however, you don't know whether that situation is acceptable or not.

By monitoring information specific to a tier, you can better determine how well your system functions. However, monitoring at the tier level might not give you the level of detail that you require.

To obtain greater levels of reporting detail, break each tier into application logical divisions, as illustrated in Chapter 1, "Introduction." Within each logical division, you can monitor metrics and events at the coarse-grained and fine-grained levels. Select which events and metrics to monitor depending on your application, but always start simply and increase complexity later.

Monitoring IIS

Monitoring IIS involves checking the functioning of the Internet protocols that the IIS service manages. However, in a Web services scenario, the World Wide Web (WWW) service is of greatest interest.

The IIS WWW service responds to HTTP requests from clients. It replies to an incoming user request and invokes the specific ASP.NET pages. ASP.NET then executes the code on the requested pages and generates the proper HTML content, which returns to the user through IIS.

The **Web Service** object contains three counters that provide useful information on the operation of IIS. For coarse-grained monitoring, you can monitor the _Total instance to monitor all of the Web sites on the computer. Table 3.1 shows these counters.

Note: Be careful using the **Web Service** object because it records information for both the HTTP and FTP protocols. The FMStocks Web site only serves up HTTP requests, so in this case you know that the values relate to work performed by the Web application. If your application has both HTTP and FTP requests being served by the same site, you should consider separating these requests by creating separate sites in IIS.

Table 3.1: IIS counters

Object: Web service, Instance: _Total	
Counter	**Description**
Current Connections	The current number of connections established with the Web service. **Threshold:** This counter depends on several variables, such as request type (ISAPI, CGI, and static HTML), CPU utilization, and so on. Only experience with running applications can provide a suitable value for this threshold.

Tables 3.2 and 3.3 on the next page show counters and describe what each counter tells you. You can collect these counters using MOM. For more information on collecting counters using MOM, see Chapter 5, "Configuring Management Applications."

Table 3.2: Standard computer counters collectible using MOM

Object: **Processor**, Instance: **_Total**	
Counter	**Description**
% Processor Time	The primary indicator of processor activity. The _Total instance is the average of all processors on a multiprocessor computer. **Threshold:** This counter should not exceed 75% for extended periods.

Table 3.3: Standard memory counters collectable using MOM

Object: **Memory**	
Counter	**Description**
Available MBytes	The amount in megabytes (MB) of physical memory available to all processes running on the computer. **Threshold:** As a guide, the value of this counter should not go below 10 MB. Keep in mind the memory consumption profile of your application.

No direct performance metrics are available for monitoring the Network Load Balancing (NLB) clustering software. You can, however, implicitly measure the performance of the NLB cluster by comparing the same set of counters, such as the % Processor Time counter, across all of the machines in the cluster. You can also use synthetic transactions to perform tests that can monitor NLB clustering, for example, by using the HTTPMon utility.

The system monitor counters in Table 3.4 add extra information to create a fine-grained picture for your IIS system. You can create a fine-grained view by monitoring each Web site instance rather than monitoring the _Total instance. Doing so allows you to separate the IIS performance metrics for each ASP.NET application.

Table 3.4: Fine-grained IIS counters

Object: **Web service**, Instance: **<specific Web site>**	
Counter	**Description**
Current Connections	The current number of connections established with the Web service. **Threshold:** This counter depends on several variables, such as request type (ISAPI, CGI, and static HTML), CPU utilization, and so on. Only experience with running applications can provide a suitable value for this threshold.

You can then add more information on processor utilization by measuring the interrupt time on the processor. This additional information is necessary because a Web server uses the processor to generate the pages, but the computer must also fetch the raw pages from disk and then send the pages to the network. Measuring the interrupt load on the computers running IIS gives you a utilization picture for the computers running the WWW service. The interrupt load counters appear in Table 3.5.

Table 3.5: Additional processor counters

Object: **Processor**, Instance: **Each processor**	
Counter	**Description**
% Privileged Time	% Privileged Time is the percentage of non-idle processor time spent in privileged mode. This counter displays the average busy time as a percentage of the sample time. % Privileged Time includes time servicing interrupts and Deferred Procedure Calls (DPCs). A failing device generating a large number of interrupts might cause a high value for this counter.
% Interrupt Time	The percentage of time the processor spends receiving and servicing hardware interrupts. This value is an indirect indicator of the activity of devices that generate interrupts. For Web servers, the primary devices are disk drivers and network interface cards.
Interrupts/Sec	The average number of hardware interrupts the processor receives and services each second. It does not include DPCs, which are counted separately.

Note: Processors running the Microsoft Windows 2000 operating system operate in one of two modes: privileged mode or user mode. Operating-system components and hardware-manipulating drivers use privileged mode, which gives direct access to hardware and all memory. Applications, environment subsystems, and integral subsystems run in user mode. Windows 2000 switches application threads to privileged mode to access operating system services.

Monitoring ASP.NET

ASP.NET is the heart of the application on the Presentation tier. It takes the raw HTTP requests from IIS, executes the code on the pages that comprise the application, and returns the dynamic pages to IIS.

A vast array of System Monitor counters exists for ASP.NET; you need only a subset for your coarse-grained health monitoring picture. These counters monitor the ASP.NET engine and can provide application-specific counters if you have more than one ASP.NET application running on the computer.

Two particularly important counters for monitoring ASP.NET are Requests Rejected and Requests Queued. Requests can be rejected for a number of reasons, such as backend latency (caused by a slow SQL server), saturation of the Web service by too many incoming requests, or a hung COM component in the Business tier.

To discover the cause of rejected requests, look at the following counters:

- **Processor/% Processor Time**
- **Process/Private Bytes**
- **Process/Thread Count**

- **ASP.NET Applications/Requests/sec**
- **ASP.NET/Requests Queued**
- **ASP.NET Applications/Pipeline Instance Count**
- **ASP.NET/Request Execution Time**

For your coarse-grained picture, you only need to monitor the __Total__ instance of the System Monitor objects you choose. This instance is the sum of the application instances. For your fine-grained picture, you need to monitor additional counters and events to increase the resolution of your coarse-grained picture.

Because of the high integration between ASP.NET and IIS, you must monitor IIS when you monitor ASP.NET, as covered earlier. ASP.NET uses the .NET common language runtime to execute the ASP.NET code, so you must also monitor the .NET common language runtime object. See the section on Monitoring .NET Common Language Runtime later in this chapter.

The System Monitor counters in Tables 3.6 and 3.7 show a coarse-grained picture of ASP.NET.

Table 3.6: Coarse-grained ASP.NET counters

Object: ASP.NET **Counter**	**Description**
Application Restarts	The number of times ASP.NET applications have restarted. An ASP.NET application restarts if ASP.NET detects a change to the application; therefore this value should be 0.
Requests Rejected	The number of requests that the ASP.NET application rejects. Applications reject requests when the Requests Queued counter exceeds the pre-configured limit. **Threshold:** The value of this counter should be zero. Investigate values greater than this using the counters listed in the previous bulleted section.
Requests Queued	The current number of queued requests for the ASP.NET application. **Threshold**: Ideally, values for this counter should be zero or near to zero. By itself, this counter does not clearly indicate operational issues. Use this counter alongside the Requests Rejected counter for a better indicator of application problems.

Note: ASP.NET performance counters reset to zero only when the **w3svc** service restarts.

Table 3.7: Coarse-grained ASP.NET application counters

Object: ASP .NET Applications, Instance: __Total__	
Counter	**Description**
Requests Executing	The number of ASP.NET requests executing in the system.
Requests/Sec	The throughput of the ASP.NET application on this computer. When Connection Requests exceeds the number of requests that can be executed, the requests queue. A full queue causes rejection of subsequent requests.

The Request Execution Time or Request Wait Time counters measure the execution and wait time of the *last* request. In applications with different kinds of requests, the time taken by the last request is not the same as the average execution time or average request wait time.

You can augment your ASP.NET monitoring by adding counters that measure usage of the ASP.NET cache. These counters are primarily designed for performance tuning, however. Instead of simply collecting the totals for all ASP.NET applications, collect data on each ASP.NET application on the system. Having the separate application instances helps you better understand which application is misbehaving.

Create a fine-grained picture by collecting the individual counters for each ASP.NET application rather than just the __Total__ instance for the ASP.NET Applications object. In addition to those counters in the coarse-grained picture, add the Pipeline Instance Count to measure backend bottlenecks as well as the cache metrics. Although the cache metrics are more appropriate for performance analysis, evaluate cache metrics in terms of the interrupt load on the computer that IIS generates. Increasing cache hits reduces the amount of disk I/O. Tables 3.8 and 3.9 show these counters.

Table 3.8: ASP.NET counters and ASP.NET applications pipeline and cache efficiency counters

Object: ASP.NET	
Counter	**Description**
Request Execution Time	The number of milliseconds taken to execute the last request. **Threshold:** Experience enables you set a threshold for an individual application. However, the counter value should be stable.

Table 3.9: ASP.NET application counters

Object: **ASP .NET Applications**, Instance: **__Total__**, each ASP.NET application	
Counter	**Description**
Pipeline Instance Count	The number of active pipeline instances. In a normally functioning application, the number of Pipeline instances is fairly constant. Sudden increases might indicate a backend bottleneck.
Cache Total Entries	The current number of both User and Internal entries in the cache. The Cache Total family of performance counters is useful for diagnosing issues with in-process session state.
Cache Total Hit Ratio	The total hit-to-miss ratio of all User and Internal cache requests.

In addition, ASP.NET generates several event log entries in the **Application** event log when certain conditions are identified with the Aspnet_wp.exe process. Table 3.10 lists event log entries that represent an error and that you should monitor.

Note: You may see ASP.NET event sources which include numbers (for example, ASP.NET 1.0.3705.0). This is because different versions of ASP.NET install newer versions of the counters. The attached numbers differ, depending on the version of the .NET Framework (and associated service packs) that you install. This gives you the choice of monitoring older counter versions.

Table 3.10: Aspnet_wp.exe counters

Event Log: **Application Event Log**, Event Source: **ASP.NET 1.0.3705.0** (or **1.0.3705.288** for SP2)	
Event ID	**Description**
1000	Aspnet_wp.exe (PID: %1) stopped unexpectedly.
1001	Aspnet_wp.exe (PID: %1) was recycled because memory consumption exceeded %2 MB (%3 percent of available RAM).
1002	Aspnet_wp.exe (PID: %1) was recycled because the number of queued requests exceeded %2.
1003	Aspnet_wp.exe (PID: %1) was recycled because it was suspected to be in a deadlocked state. It did not send any responses for pending requests in the last %2 seconds. **Note:** The deadlock detection mechanism in ASP.NET recycles the Aspnet_wp.exe if the ASP.NET worker process has been idle for the time specified for the responseDeadlockInterval configuration setting in the Machine.config file.

Monitoring .NET Common Language Runtime

The .NET common language runtime compiles the ASP.NET code for execution. The common language runtime also provides a number of its own counters, but you only need a few to create your coarse-grained health picture. The main concerns for monitoring at this level are that memory is being properly used (no memory leaks or excessive garbage collection) and that the level of exceptions being generated by the application (on a percentage basis) does not overload the CPU. Table 3.10 shows the counters to monitor so that you can determine whether these concerns are a problem.

To detect whether there is a memory leak, watch the process Private Bytes size. Watching Private Bytes is a time-honored tradition (but not a military one) for detecting memory leaks, and still works well with .NET-based applications.

In addition to watching for memory leaks, the .NET Framework provides .NET-based applications with built-in memory management using garbage collection. Garbage collection runs on a regular basis and reclaims application memory. A .NET-based application can misbehave in that the application can cause additional system overhead through excessive garbage collection. Although some garbage collection is necessary for a healthy application, Gen 2 garbage collection is very expensive in terms of system resources and should be minimized wherever possible. This type of garbage collection suspends application threads.

The best way to determine whether your application is running within prescribed limits is by measuring the Gen 2 collections as a percentage of the total collections. Use of a percentage figure automatically accounts for differences in loading levels, making light loads and heavy loads directly comparable.

When garbage collection is doing its job, you will see a saw-tooth pattern to the values for the Private Bytes counter within clear upper and lower boundaries. If your application has a memory leak, you will notice a steady upwards trend for each upper boundary, although this might be difficult to spot with a small memory leak. When the application hits the recycle threshold, you will see the counter value drop steeply to zero and then climb again as the worker process recycles. An entry in the Application log informs you of the recycle event.

Another aspect of a healthy .NET-based application is the frequency of application exceptions. If you measure application exceptions as a percentage, then you can compare light and heavy loads. A healthy application has an exception rate of less that 5 percent of the total requests processed. Exception processing is disproportionately expensive compared to normal operations, so if you measure rates higher than 5 percent, track down and resolve the causes of these exceptions.

Note: Some application instructions generate exceptions as part of normal processing. For example, **Response.Redirect**, **Server.Transfer**, and **Response.End** all generate multiple occurrences of the **ThreadAbortException**.

Table 3.11: Counters monitoring the .NET common language runtime

Object: .NET CLR Memory, Instance: aspnet_wp, aspnet_state	
Counter	**Description**
% Time in GC	The percentage of elapsed time spent performing a garbage collection (GC) since the last GC cycle. Keep this number small and nonzero for short periods of time. Excessive GC time indicates an inefficient program.
# Bytes in all Heaps	The number of bytes currently allocated in all of the managed heaps. This value is the sum of Gen 0 Heap Size, Gen 1 Heap Size, Gen 2 Heap Size, and the Large Object Heap Size.

Table 3.12: Counters monitoring .NET common language runtime exception rate

Object: .NET CLR exceptions, Instance: aspnet_wp, aspnet_state	
Counter	**Description**
# Exceps Thrown/sec	The total number of exceptions generated per second in managed code. Exception handling by the common language runtime is very expensive and can indicate possible application problems. As this value increases, performance degrades. **Threshold:** This counter value should be less than 5 percent of the value for Request/sec for the ASP.NET application. Once you have gained experience operating an application, you can set new threshold values.

Table 3.13: Counters for the aspnet_wp process

Object: Process, Instance: aspnet_wp	
Counter	**Description**
Private Bytes	The current number of bytes allocated to this process that cannot be shared with other processes.
Thread Count	The number of threads currently active in the aspnet_wp process.

There are no additional metrics to monitor for your fine-grained picture.

For more information on common language runtime performance counters in the .NET Framework, see "Performance Counters" on the MSDN Web site at *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpgenref/html/gngrfperformancecounters.asp.*

Monitoring .NET Remoting

.NET Remoting uses HTTP as its transport protocol, so, just like the Presentation tier, NLB can provide load balancing for the Business tier. Hence you can monitor the same information in your Business tier. Also monitor the **Web Service** object, the **ASP.NET** objects, and the **.NET CLR** objects as you did in the preceding section.

You can use the counters for Context-Bound Objects Alloc/Sec and Remote Calls/Sec to see how noisy the interface between the Presentation tier and Business tier is, as Table 3.14 shows. An excessively "chatty" interface degrades application performance. For the FMStocks application, you should monitor the managed code components.

Table 3.14: Monitoring events through .NET CLR Remoting

Object: .NET CLR Remoting, Instance: aspnet_wp	
Counter	**Description**
Context-Bound Objects Alloc/Sec	The number of context-bound objects allocated per second.
Remote Calls/Sec	The number of remote procedure calls invoked per second.

Monitoring Managed Code Components

Many middle-tier applications consist of business logic components encapsulated within reusable modules. Components of this type that are written using the .NET Framework are called *managed code components* because the .NET Framework common language runtime manages their execution. FMStocks is an example of an application built from managed code components and you might already be running similar components in your own environment.

How you monitor these managed code components depends on whether they are serviced components—that is, they use Enterprise Services. If your managed code components do not use Enterprise Services, your developers need to add suitable code to produce the required monitoring data. For more information about the best way to add this code, see Chapter 4, "Instrumenting .NET-Based Applications."

There are several tools that allow you to look inside the common language runtime GC heap and understand the objects allocated from your application. One example is Allocation Profiler, which is available on the GotDotNet Web site at *http://www.gotdotnet.com/Community/User/Samples/download.aspx?FileGuid=6066e20c-c159-428c-89e4-92c42397a8d8.*

You can obtain the Debugging .NET Applications prescriptive architecture guide from the MSDN Web site at *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/DBGrm.asp.*

Monitoring Serviced Components

In the .NET Framework, you can write serviced components using managed or unmanaged code. Many of the applications currently running in the data centers use unmanaged code serviced components, otherwise known as *COM+ components.* Managed code serviced components are created using the .NET Framework and use all of the same services that traditional COM+ components do. Both types run under Component Services (COM+).

Monitoring individual components allows you to break the work being done by your application into the types of transactions running inside the application. This breakdown lets you better understand the workload distribution both between applications and within an application. For example, you can discover which applications are most active during certain periods of the day.

Serviced components run in processes named *Dllhost.exe.* For coarse-grained monitoring, you can monitor individual Dllhost.exe processes. The bad news is that you might encounter several Dllhost.exe processes, and it is difficult to sort out which application belongs to which process.

For the coarse-grained view, monitor the counters listed in Table 3.15.

Table 3.15: Monitoring serviced components through Dllhost.exe

Object: Process, Instance: DLLHOST	
Counter	**Description**
% Processor Time	The primary indicator of application loading on the processor. Note that the **_Total** instance does not provide any useful information in this case. **Threshold:** This counter should not exceed 70% for extended periods.
Private Bytes	The current number of bytes allocated to this process that cannot be shared with other processes.

For fine-grained monitoring, you need information on which Dllhost.exe process is associated with each application and statistics about specific components. This information can be difficult to acquire, however, because most data centers have many COM+ applications. For example, a medium-sized data center might have 50 to 100 COM+ application processes, and a large data center may have 200 to 300 application processes. Efficiently monitoring these application processes and the components within them is key to a monitoring your .NET-based application.

Enterprise Services provides you with a complete set of metrics to monitor the functioning of serviced components. But these metrics are difficult to collect because they require a tremendous amount of knowledge on how the service component infrastructure works as well as being frightfully complex to correlate. To save time, you can use a product such as AppMetrics for Transactions from Xtremesoft to collect these COM+ system metrics and let it do all of the hard work. Table 3.16 shows some of these metrics.

Table 3.16: COM+ system metrics

Application/package: ***Each application process on your system***	
Metric	**Description**
Active	The COM+ Application status (1 = started; 0 = stopped).
CPU	The processor time consumed by the COM+ application process.
Crash	The number of times the COM+ application process stopped because of an unhandled exception.
Fault	The rate in which page faults occur in the threads executing in the COM+ application process.
Name	The name of the COM+ or Enterprise Services application executing within the process.
Mem	The current number of bytes in the working set of the COM+ application process.
Start	The number of times the COM+ application process started.
Stop	The number of times the COM+ application process shut down, either by exceeding its *Idle time* parameter or manually by the operator.
Thread	The number of threads in the COM+ application process.
Virtual Bytes	The current size in bytes of the virtual address spaced the COM+ application process uses.

Identifying each process by application name—rather than trying to cope with hundreds of identical Dllhost.exe processes—means that you can identify which applications are consuming resources excessively by application name. Examples of excessive resource consumption are memory leaks or runaway processes.

Each component class running Enterprise Services can be monitored individually using AppMetrics for Transactions. AppMetrics provides you with the information to construct a fine-grained picture that separates work into individual transactions, such as Table 3.17 shows.

Table 3.17: Individual transaction information from Enterprise Services

Component: ***Each serviced component class***	
Metric	**Description**
Aborted	The number of component instances that aborted during the last interval.
Active: Begin	The number of component instances that were active when the last interval began.
Active: Maximum	The peak number of active component instances during the last interval.

(continued)

Component: *Each serviced component class (continued)*	
Metric	**Description**
Active: Minimum	The minimum number of active component instances during the last interval.
Application	The name of the COM+ or Enterprise Services application.
Completed	The number of component instances that completed during the last interval.
Component	The COM Prog ID for the component.
Duration: Average	The average duration in milliseconds of component instances that completed during the last interval.
Duration: Maximum	The maximum duration in milliseconds for component instances that completed during the last interval.
Duration: Minimum	The minimum duration in milliseconds for component instances that completed during the last interval.
Rate: Aborted	The rate per second at which component instances abort.
Rate: Completed	The rate per second at which component instances complete.
Rate: Started	The rate at which components started during the last interval.
Started	The number of component instances that started during the last interval.
This Session: Current	The current number of active component instances since the monitor started.
This Session: Maximum	The maximum number of active component instances since the monitor started.
This Session: Minimum	The minimum number of active component instances since the monitor started.

In addition to these metrics, AppMetrics for Transactions provides you with several alerts with thresholds to help you "manage by exception," that is, identify the transactions that do not complete and thus generate exceptions. These alerts are on a per component class basis. This facility makes it easy for you to let AppMetrics watch for runaway or hung components by monitoring the events listed in Table 3.18.

Note: For an explanation of managing by exception, see Chapter 4, "Instrumenting .NET-Based Applications."

Table 3.18: Process alerts for Enterprise Services

Process alerts: *<COM+ application name>*	
Alert	**Action**
Percent CPU	Send an alert if the %CPU utilization for this application process exceeds the specified threshold.
Threads	Send an alert if the number of application process threads exceeds the specified threshold.
Page Faults Per Second	Send an alert if the number of page faults for this application process exceeds the specified threshold.
Virtual Bytes	Send an alert if the virtual byte count for this application process exceeds the specified threshold.
Working Set	Send an alert if the memory working set for this application process exceeds the specified threshold.

Component alerts provide an aggregate value for all of the component instances that start, complete, or abort during a specified time interval. Generally, this interval is between 1 and 5 minutes. Setting alerts for the average duration is an excellent way to detect hung components. Table 3.19 lists some typical component alerts.

Table 3.19: Component alerts

Component alerts: *<serviced component class>*	
Alert	**Action**
Number Aborted	Send an alert if the number of component instances that abort exceeds the specified threshold.
Number Started	Send an alert if the number of component instances started exceeds the specified threshold.
Average Duration	Send an alert if the average duration of a component instance exceeds the specified threshold.

AppMetrics Diagnostics Monitor

For even finer-grained monitoring that is primarily useful for analyzing and trouble-shooting performance, AppMetrics for Transactions uses Diagnostics Monitor. Diagnostics Monitor records details about individual component and transaction activity in a running application.

Diagnostics Monitor displays its metrics in several reports, but the Drill-Down Report provides the most value in diagnosing application problems. This report shows durations for each active component in the application. It also shows the logical chain of method calls between the components. This is important because merely viewing the metrics for each component in isolation from other components does not tell the whole story. In particular, it provides no information about the calls that a component makes to other components.

When you have the information about the chains of method calls between components, you can begin to see the relationships between components during runtime. That information lets you analyze the overall response time of a component based on its constituent parts. From this analysis, you can determine whether a bottleneck occurs either in the root component object or somewhere further down the method call chain in a different component.

Figure 3.5 shows an excerpt from an AppMetrics for Transactions Drill-Down Report that reveals the logical chain of method activity for a single call sequence in FMStocks. Figure 3.5 shows the following areas:

- FMStock7.DAL.Broker component invokes a subordinate FMStocks7.GAM.7 component instance
- The **CreditAccountBalance** method, which was invoked on the subordinate component
- The unique naming convention for cross-application calls

The columns in Figure 3.5 display the transaction name and the methods that the transaction calls. The Rel Start and Rel End columns record the execution time in milliseconds from the beginning of the FMStocks7.DAL.Broker transaction. The Duration column records the time taken to complete the transaction or a component of that transaction.

When a transaction makes a call to another application, the cross-application call includes the other application name plus a colon. In Figure 3.5, the FMStocks7.GAM.7 component resides in a different application from the FMStocks7.DAL.Broker component.

Figure 3.5 shows that the excessive execution time for the FMStocks7.DAL.Broker transaction results from the time the **CreditAccountBalance** method took.

Transaction	Method	Rel Start (ms)	Rel End (ms)	Duration (ms)
FMStocks7.DAL.Broker		0	9960.599609	9960.599609
	FMStocks7.DAL.Broker->GetComponentInfo	2.712890625	2.78125	0.068359375
	FMStocks7.DAL.Broker->GetObjectIdentity	4.42578125	4.443359375	0.017578125
	FMStocks7.GAM : FMStocks7.GAM.7->CreditAccountBalance	72.74416063	7354.029297	7281.285156

Figure 3.5
AppMetrics for Transactions Drill-Down Report

For more information about the AppMetrics for Transaction Drill-Down Report, see Chapter 6, "Notifying and Reporting."

The alerts from AppMetrics can be sent to MOM for handling. For information on how to configure MOM to receive alerts from AppMetrics, see Chapter 5, "Configuring Management Applications."

Monitoring the Data Tier

Monitoring the Data tier means monitoring your database management system. You must include information on the Data tier to create a complete end-to-end picture of your application. The FMStocks Database tier uses SQL Server, although you might use another database management system for your enterprise.

Monitoring SQL Server is far too large a topic to cover in this book. Fortunately, the documentation that comes with SQL Server contains loads of useful information on this subject. The following topic, however, analyzes the main areas to consider when building your coarse-grained picture. For more information that you can use to construct your fine-grained Data tier picture, consult SQL Server online Help.

Note: When monitoring SQL Server, you must monitor both the performance of the underlying system and the SQL Server-specific counters and events.

The coarse-grained picture for the Data tier contains more subdivisions than in the Web tier or the Business tier because the Data tier is the biggest influence on overall system performance and health.

Processor utilization and physical memory utilization are part of a very coarse-grained picture because they are the primary health indicators on any computer. For the same reason that interrupts are important on the Presentation tier, monitoring them is equally important in the Data tier. This is because the computer running the database management system must use the processor to fetch records and to carry out sorts. Tables 3.20 and 3.21 display these counters.

Table 3.20: System processes for monitoring the Data tier

Object: Processor, Instance: _Total	
Counter	**Description**
% Processor Time	The primary indicator or processor activity. The _Total instance is the average of all individual processors on a multiprocessor computer. The value is between 0 and 100.
% Privilege Time	The percentage of non-idle processor time spent in privileged mode. This counter displays the average busy time as a percentage of the sample time. % Privileged Time includes time servicing interrupts and DPCs.
% Interrupt Time	The percentage of time the processor spends receiving and servicing hardware interrupts. This value is an indirect indicator of the activity of devices that generate interrupts. For the Data tier, the primary devices are disk drivers and network interface cards.
Interrupts/Sec	Interrupts/Sec is the average number of hardware interrupts the processor receives and services each second. It does not include DPCs, which are counted separately.

Table 3.21: System memory counters for monitoring the Data tier

Object: **Memory**	
Counter	**Description**
Available Mbytes	The amount of physical memory in MB available to processes running on the computer. An alert occurs when the value goes below a threshold value — generally, 10 MB. By default, SQL Server takes advantage of available memory, although you can limit the amount requested.
Pages/sec	The number of hard page faults per second. A hard page fault occurs when a process accesses a page that is not loaded in physical memory. In this situation, the Virtual Memory Manager has to fetch the relevant page or pages from the hard disk. Hard page faults can cause system-wide delays and decrease the speed of information retrieval by a factor of 1,000.

The final Data tier counter is for the Working Set of the SQL Server process. Table 3.22 illustrates this counter.

Table 3.22: SQL Server Working Set counter for monitoring the Data tier

Object: **Process**, Instance: **sqlservr**	
Counter	**Description**
Working Set	The number of bytes that the SQL Server process is currently using.

Because SQL Server is connected to the rest of the application through the network, monitoring network activity provides a key indicator of system health. Table 3.23 shows network monitoring of the Data tier through the Bytes Total/sec counter, which is the most commonly used network counter for monitoring network activity.

Table 3.23: Data tier network monitoring

Object: **Network Interface**, Instance: ***<each network card>***	
Counter	**Description**
Bytes Total/sec	The number of bytes in and out of each network card. Low rates of network activity during high application traffic indicate general network congestion.

For the Data tier, disk performance is crucial even for the most basic health monitoring. It is worth monitoring each data drive separately. Coarse-grained monitoring records only the transfer rate, as Table 3.24 shows, but you can get much fancier monitoring scenarios based on your application's transaction patterns.

Table 3.24: Physical disk monitoring in the Data tier

Object: **Physical Disk**, Instance: ***<drives containing data>***	
Counter	**Description**
Disk Transfers/sec	The transfer rate for both read and write transactions.

SQL Server-specific metrics help you understand how the resources described above (such as processor, memory, and so on) translate into SQL Server operations. Tables 3.25 through 3.29 show some of the counters that you can use.

Table 3.25: SQL Server access metrics

Object: **SQLServer:Access Methods**	
Counter	**Description**
Full Scans/sec	The rate of full table or full index scans. Lower numbers are better.

Table 3.26: SQL Server buffer manager metrics

Object: **SQLServer:Buffer Manager**	
Counter	**Description**
Buffer Cache Hit Ratio	The percentage of pages retrieved from the SQL Server buffer pool without retrieving the page from the hard disk. An efficiently running system will have a high Buffer Cache Hit Ratio and low amount of Physical Disk Reads.

Table 3.27: Per instance SQL Server database metrics

Object: **SQLServer:Databases**, Instance: ***<each application database>***	
Counter	**Description**
Log Growths	The number of transaction log growths. Primarily used for capacity planning.
Percent Log Used	The percentage of the transaction log file currently in use. Used in conjunction with Log Growths.
Transactions/sec	The rate of completing transactions. This is the primary indication of database throughput.

Table 3.28: SQL Server lightweight lock metrics

Object: **SQLServer:Latches**	
Counter	**Description**
Average Latch Wait Time (ms)	The average amount of time that a request for a latch had to wait within the database. Latches are lightweight, short-term row locks, so higher numbers indicate contention for resources.

Table 3.29: SQL Server lock metrics

Object: SQLServer:Locks, Instance: _Total	
Counter	**Description**
Average Wait Time (ms)	The amount of time that a request had to wait to fulfill a lock request. Lower times are better; higher values indicate resource contention.
Lock Waits/sec	How long lock requests must wait to be satisfied, expressed as a rate. Lower values are better.
Number of Deadlocks/sec	The number of lock requests that turn into deadlocks, expressed as a rate. Lower values are better.

Monitoring SQL Server with MOM using the SQL Server Management Pack makes monitoring SQL Server very easy. MOM provides some useful alerts for monitoring the health of your SQL Server as well as for monitoring any of the metrics in the preceding section. These alerts—shown in Table 3.30—allow you to prevent problems from bringing down your application.

Table 3.30: SQL Server monitoring with MOM

SQLServer alerts	
Alert	**Action**
DBSpace	Send an alert if disk space in the database is low. Do not use this counter if the database is set to **Automatically grow file**, as it will generate unnecessary alerts.
DevSpaceAvail	Send an alert if physical disk space is low.
LogSpace	Send an alert if log space is low. Optionally, truncate the log file.
NearMaxConnect	Send an alert if SQL Server is running low on available database connections.
NearMaxLocks	Send an alert if SQL Server is almost out of locks.

For more information on monitoring SQL Server, see:

- Microsoft SQL Server Resource Kit available on the Microsoft Web site at *http://www.microsoft.com/sql/techinfo/reskit/default.asp*
- SQL Server 2000 Operations Guide available on the Microsoft Web site at *http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/sql/maintain/operate/opsguide/default.asp*

Monitoring Using Synthetic Transactions

Examining application workload by monitoring processor usage, disk usage, network traffic, and so on provides useful data, but this type of monitoring is inherently limited in what it can tell you about the health of your application. This kind of monitoring requires that users do real work with the application, and user loads rarely follow predictable patterns.

To create predictable usage patterns, implement some form of monitoring that replicates user transactions, and verify that these transactions complete correctly. These are called *synthetic transactions.*

A synthetic transaction is a system or an application transaction performed regularly to test a service. Synthetic transactions create a known load, which you can then monitor.

Synthetic transactions complement the metrics and events you collect by alerting you to errors of omission. Here you detect not activity, but lack of activity. For example, if you see no usage of your system, synthetic transactions can tell you whether that is because no one is using your service or because, say, the network card to the outside world has just died. If synthetic transactions are coming into your system, you can tell the difference because either the agent driving the synthetic transaction sends an alert (because it can't complete the transaction) or MOM records the omission of the transaction event.

You can use different levels of complexity and sophistication in your synthetic transaction, similar to the concepts of coarse-grained and fine-grained monitoring. You can, for example:

- Build special transactions into the application.
- Use commercially available programs, shareware, or even freeware tools to create synthetic transactions.
- Use third-party applications available from a number of Microsoft partners.

An example of using third-party applications is a program that replays the posts from a user to the Web site. This application then checks the returned content at each stage. Third-party service providers can implement this type of monitoring and notify of errors or alerts.

Another technique involves using a network protocol analyzer to check total traffic to the site, so you can see the contents of the network packets and the number of timeouts—404 or 500 IIS errors—that users receive.

Note: Connections using Secure Sockets Layer (SSL) encryption limits this technique, as the packet contents will be encrypted.

It takes extra work to build such transactions. You must also be careful about creating security holes or points of attack. Security holes can arise from the guest user accounts for the synthetic transactions, and exposing testing points to the Internet can invite attacks. Also, synthetic transactions may affect your business reporting metrics.

Note: You can use security measures, such as running the synthetic transactions from a known fixed IP address and properly filtering packets from that client, to allow the synthetic transactions to complete.

You can download Web Monitor, the freeware testing tool, from the MSDN Web site at *http://msdn.microsoft.com/code/default.asp?URL=/code/sample.asp?url=/MSDN-FILES/026/001/215/msdncompositedoc.xml.* Alternatively, you can use Cluster Sentinel, which is part of the Windows 2000 Server Resource Kit. For more information on Web Monitor and step-by-step procedures for using this utility, see the sections on Web Monitor and Customizing Web Monitor in Chapter 5, "Configuring Management Applications".

In the Fitch and Mather example of synthetic transaction monitoring, the following factors are important:

- Monitoring must be external to the application (no additional code).
- Testing must include checking Web site availability.
- Fitch and Mather does not want to pay a third party for testing.

To monitor the application, Fitch and Mather use a simple monitor for their Web servers using Web Monitor. This tool runs as a scheduled task and probes the main page for the FMStocks Web site. If returning that page fails, Web Monitor writes an event log entry that MOM picks up.

Fitch and Mather chose to use the Web Monitor with FMStocks because Web Monitor is simple and easy to configure. Web Monitor checks that the Web server is running and can carry out simple validation of page content. Although returning just the FMStocks home page does not go very deep into the application, Web Monitor provides a good starting point.

An enhancement of synthetic transactions is end-to-end monitoring. An end-to-end monitoring tool can replay user transactions at the front end and then check that the correct database records appear in the Data tier.

Summary

In this chapter, you looked at the monitoring process using system counters to achieve a coarse-grained or fine-grained picture of your application. You saw how you can use the concepts of tiers and logical divisions to partition an application. Finally, you looked at synthetic transactions that can provide varying levels of application monitoring.

4

Instrumenting .NET-Based Applications

Material in this chapter is dependent on the release of Windows .NET Server 2003, and the PDF will be updated once the product is released. Please check the Web site for updates after the release of Windows .NET Server 2003. We apologize for any inconvenience this may cause.

5

Configuring Management Applications

Introduction

You can capture both system monitoring and instrumentation data using management applications such as Application Center 2000, AppMetrics for Transactions from Xtremesoft Inc., and Microsoft Operations Manager (MOM).

This chapter reviews the physical design and the architecture of the management solution. It then covers monitoring techniques using Application Center 2000, AppMetrics, and MOM.

Throughout this chapter, you employ the concepts of logical divisions, tiers, and coarse-grained and fine-grained monitoring, as described in Chapter 2, "Application Monitoring Concepts." These concepts separate the different structural areas within a typical distributed .NET-based application, such as the sample FMStocks application that this book uses. For more information about the Fitch and Mather data center scenario, see the Introduction to this book.

Note: The planning, installation, and configuration of Application Center 2000, AppMetrics, and MOM are outside of the scope of this document. See the References section at the end of this chapter for links to information about setting up and configuring these applications.

Implementing a Management Architecture

As a business or organization grows, so does the complexity of its information technology (IT) infrastructure. This complexity taxes even the most experienced support teams, making automated, intelligent, and proactive monitoring solutions essential.

To assist with this challenge, Microsoft developed a suite of core management services incorporated into the base operating system:

- Event logging
- Performance counters
- Windows Management Instrumentation (WMI)
- Event trace logging

These core services provide the instrumentation and interfaces both to manage the Windows operating system (and associated programs) and to develop management applications.

Selecting Management Applications

Microsoft and a number of certified partners supply a family of management applications that use the core technologies to provide enterprise-level IT management services and solutions. Management applications also gather and collate monitoring information from the core management services.

A management solution for a .NET-based application might include the following components:

- Application Center 2000
- AppMetrics for Transactions
- Synthetic Transactions such as Web Monitor
- Microsoft Operations Manager

This chapter covers these management applications in later sections. Figure 5.1 shows the physical implementation of the Fitch and Mather management environment.

Note: FMStocks uses .NET Remoting over HTTP with binary encoding for communication between the Presentation tier and the Business tier. The term DCAM is a combination of Data Access Server (DAS) and Consolidator and Agent Manager (CAM).

The MOM database server exists in the Data tier on the assumption that you manage all of the SQL Servers similarly. However, your organization may put the SQL Server running the MOM database in the management environment.

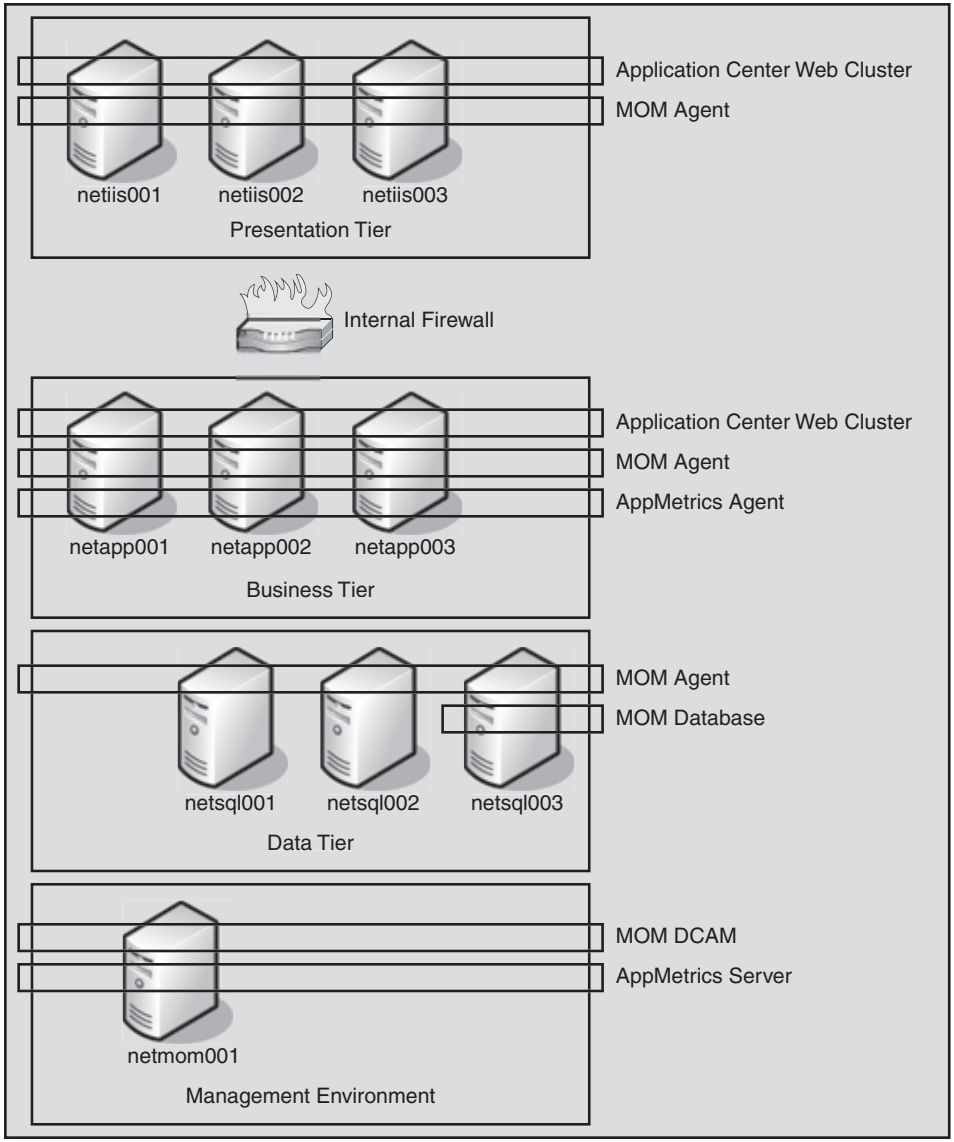


Figure 5.1
Fitch and Mather management solution

Application Center 2000

Application Center 2000 is an integral component of the Microsoft .NET platform and is the primary platform for building and deploying Web-based solutions. Application Center 2000 enables scalable and reliable Web applications at both the Presentation tier and the Business tier through software scaling. Application Center 2000 does the following:

- Provides software scaling
- Simplifies Web and component application management
- Increases Web and component application availability

Application Center 2000 allows you to implement and manage the Presentation and Business tier clusters, provides a coarse-grained view of the nodes (computers) within the clusters, and enables monitoring of the clusters as a unit. Application Center includes powerful features to view performance metrics and event log data for one cluster node or for an entire cluster. You can even monitor applications remotely using a browser-based console interface.

Application Center 2000 combines existing tools, such as Health Monitor and Event Viewer, with its own tools to provide cluster monitoring. This combination allows you to deal effectively with cluster health and performance issues.

Although this book adopts the concept of dividing physical tiers into logical layers, it is difficult to apply this concept to Application Center. Application Center manages Web and component clusters, and there is great deal of integration between Internet Information Services (IIS), COM, Network Load Balancing (NLB), and Application Center. From this perspective, it is easier to take the tier view when working with Application Center.

By default, Application Center 2000 gives you a coarse-grained view of your IIS system. This view includes IIS, COM, NLB, and Application Center tasks, such as deployments.

Using Application Center, you can augment your coarse-grained monitoring picture to include application-related information. With a larger investment of effort and time, you can increase the resolution of your system and application monitoring by adding performance counters to monitor both system and application health. This approach applies to organizations that do not use an enterprise management system such as MOM.

Supporting Clusters

Application Center 2000 supports Web clusters and Component Load Balancing (CLB) clusters. Web clusters support TCP/IP-based communication and CLB clusters support Component Object Model (COM) or Distributed COM (DCOM)-based communications.

Determine which cluster type you need based on the communication protocol between servers and clients. With Application Center 2000, you normally use CLB clusters to load balance and create redundancy between the Presentation tier and the Business tier (assuming that you have implemented these as two physical tiers). Web clusters run the client-facing Web sites and ASP.NET-based applications.

Using Application Center with .NET Remoting

With .NET Remoting, you can use an Application Center Web cluster to load balance the Business tier because you can configure .NET Remoting to communicate over HTTP running on TCP/IP.

In our example scenario, the Fitch and Mather environment consists of three physical tiers and uses .NET Remoting to connect the Presentation tier to the Business tier.

Note: The FMStocks test environment deliberately exposes .NET Remoting in this manner. Assess your own design to see whether this approach is appropriate.

Viewing Performance Counters

Application Center provides a set of performance monitoring counters that you can use to monitor by cluster or by cluster member. You can view these default counters by clicking the **Add** button on the bottom right of the Application Center console. This button then allows you to add counters to the Performance graph in the Application Center console.

In addition to these default performance monitor counters, you can add counters by writing a counter definition as a Managed Object Format (MOF) file and compiling it with the MOF compiler Mofcomp.exe. As a rule of thumb, add extra counters only if they are essential. Use other tools, such as Performance Monitor, for short-term requirements, such as fine-tuning or troubleshooting.

Note: The Application Center installation compact disk has an MOF sample file to add a Performance Monitor counter. See the Application Center 2000 Resource Kit for more information.

Using Health Monitor

Health Monitor Version 2.1 is an Application Center component that monitors specific tasks and generates notifications on exceeding predefined thresholds. Health Monitor consists of two components:

- Health Monitor Console
- Health Monitor Agent

During installation, you can install either or both of these components on the local system. If you log on as an administrator, you can add computers and edit monitoring configuration settings through the Application Center console. The Application

Center agent then gathers data through its data collectors, tests for threshold violations, and generates alerts.

The Health Monitor Agent both provides and consumes WMI events. The agent ships with several providers, including:

- HTTP
- COM+
- Core Agent
- PING
- TCP/IP Port Connect

In this chapter, you look at the HTTP and COM+ providers in detail, as these providers are the most useful for monitoring .NET-based applications.

Health Monitor lets you define actions and associate them with a data group or data collector. Application Center has five default actions, which the "Actions Node" section later in this chapter covers. You can create specific actions based on your application, your environment, your requirements, and so on.

The Health Monitor Version 2.1 user interface consists of the following nodes for each monitored computer.

- **Actions node—**Stores and manages actions associated with specific monitors.
- **Non-Synchronized Monitors node—**Contains monitors that you can configure for use on individual computers. These monitors do not replicate across a cluster.
- **Sample Monitors node—**Lets you install a collection of sample monitors created for demonstration purposes. You can then customize these samples to suit your own cluster environment.

> **Note:** If you chose the typical Application Center setup when you installed the software, the Sample Monitors node is empty. You can always add the sample monitors later by running the Application Center setup program again.

- **Synchronized Monitors node—**Works across the cluster, and replicates these monitors to every computer in the cluster.

Figure 5.2 shows these four nodes within Health Monitor.

Application Center installs a collection of synchronized monitors by default. These monitors fall into the following categories:

- Application Center Monitors
- Online/Offline Monitors
- System Monitors
- Web Site Monitors

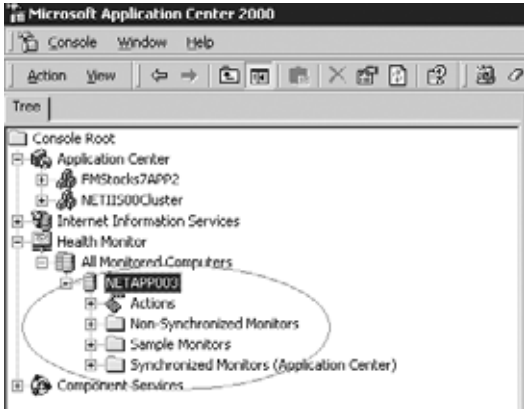


Figure 5.2
Health Monitor user interface

Actions Node

The **Actions** node lets you configure actions that occur after receiving a particular event. The default actions installed by Application Center are:

- **Take server online**
- **Take server offline**
- **Log to offline.log**
- **Log to websitefailures.log**
- **Email Administrator**

Additionally, you can create the following actions:

- **Command Line Action**
- **Text Log Action**
- **Email Action**
- **Script Action**
- **Windows Event Log Action**

Normally, you create your actions before you create your monitors and then associate each monitor to a specific action. However, you can create an action after you create a monitor and then edit the monitor's properties to associate it with the action.

HTTP Provider

The HTTP Provider supports the required interfaces for exposing instances of WMI provider services. The HTTP Provider monitors HTTP requests and responses using WMI and provides statistics to a monitoring tool (such as Health Monitor) on Web application availability and performance.

Through the HTTP Provider, Application Center uses Health Monitor to execute HTTP requests and receive responses. This functionality enables you to monitor Web application performance and availability programmatically. You can then direct the computer to perform specified actions based on the received information, such as stopping and starting a service.

COM+ Provider

Application Center's COM+ Provider delivers simple statistics on the performance of COM+ applications, such as transactions per second and how many times an application shut down because of errors. This provider compensates for the lack of performance monitoring counters for COM+ objects and the lack of a WMI provider for instrumentation.

You can configure the COM+ provider to create a notification if a monitored system meets or exceeds predefined thresholds. The provider gives you information that is not readily available, such as failure shutdowns, object activations, or committed transactions. The provider lets you select COM+ applications to monitor and customize the amount of monitoring data generated. This provider lets you monitor all the COM+ objects and event information for an application with minimal processing overhead.

For example, you can create a COM+ collector to monitor a specific COM+ application for issues such as aborted transactions per second, administrative shutdowns, failure shutdowns, and object activations per second. This monitor can then notify you if a particular COM+ application shuts down or reaches a set threshold on the number of activations.

Note: Application Center 2000 does not support the COM+ Library applications that FMStocks uses. The limitation lies with the implementation of COM+ in Windows 2000, which does not provide information to WMI about COM+ Library applications. This book does not provide instructions on how to set up the COM+ monitor with FMStocks.

AppMetrics for Transactions

AppMetrics for Transactions monitors COM components running on Windows 2000 Serviced Components as either managed or unmanaged code. Most importantly, AppMetrics lets you see individual application dependencies, providing detailed reports on the performance of individual applications running within the Business tier together with the performance of each dependent component. This enables detailed tracking of individual issues within applications, particularly important in a multi-tier data center environment running possibly hundreds of line-of-business applications.

AppMetrics complements MOM by providing MOM with otherwise unavailable information about serviced components. AppMetrics monitors serviced components without code instrumentation, so it is a good choice for producing a fine-grained picture of your application. This feature is useful when you cannot alter the source

code, because the application is already in production or because you purchased the component from an external company.

AppMetrics assists you with the overall management of serviced components running in the Business tier. AppMetrics generates metrics on the health of your .NET-based components, whether those components use managed or unmanaged code.

The easiest way to get the most out of the combination of MOM and AppMetrics is for AppMetrics to pass the component and application process alerts to MOM. The following topics describe how to configure AppMetrics to generate alerts and then configure MOM to monitor for those alerts.

Synthetic Transactions

One of the advanced monitoring techniques is using synthetic transactions. Synthetic transactions monitor operations from the service user's points of view. This technique does not replace any of the other methods, such as using system counters, but it enhances the overall picture of your .NET-based application. Examples of this type of application include numerous shareware, freeware, or commercial programs, such as Cluster Sentinel (part of the Application Center Resource Kit).

Web Monitor

Web Monitor is a shareware monitoring tool that you can use to generate synthetic transactions and is suitable for relatively simple applications or intranet applications. Web Monitor can remove a node from an NLB cluster when the Web server on that computer fails to respond. You can also use Web Monitor to distinguish between a Web server that is not functioning and one that is unreachable due to connectivity problems.

For more information on Web Monitor, see "A Simple XML-driven Tool: Monitor Your Web Site's Activity with COM and Active Scripting" on the MSDN Web site at *http://msdn.microsoft.com/msdnmag/issues/0700/webmon/webmon.asp*.

Web Monitor Architecture

Web Monitor consists of six files that must exist in the same directory: four Microsoft JScript® development software files (main.js, startServices.js, httpMonitor.js, and mail.js), an XML schema file (monitorSchema.xml), and a Windows script file. You configure Web Monitor through an XML file called URLStomonitor.xml, which you include as a command line argument when starting Web Monitor. See the steps in "Customizing Web Monitor" later in this chapter for the correct syntax.

The following steps represent the process Web Monitor follows when monitoring your .NET-based application:

1. Web Monitor reads the XML configuration file included in the command-line argument.

2. Web Monitor parses this XML file and validates the syntax against the provided XML schema.
3. For every Web site in the file:
   a. Web Monitor sends a request to the Web site.
   b. If the site functions correctly, Web Monitor proceeds to the next Web site.
   c. If the Web site does not respond, Web Monitor sets a status variable to false.
4. If a Web site does not respond, Web Monitor can:
   a. Take corrective action.
   b. Send e-mail to notify administrators.
5. Web Monitor records events in the event log.

Note: You can automate monitoring with Web Monitor using the Windows Task Scheduler.

Microsoft Operations Manager

MOM provides comprehensive event management, proactive monitoring, notifications, reporting, and trend analysis in large, distributed environments. MOM also provides a centralized management console, extracting and presenting data from other management applications. You can use additional management pack modules to extend the capabilities of MOM and create a fine-grained view of all three tiers.

MOM collects data from several providers, such as events, counters, and WMI. MOM captures all this information using rules and either responds to a specific fault scenario with a predefined action or consolidates the data into a more meaningful event. Automated responses to such events can range from sending e-mails or paging calls to triggering preprogrammed remedial actions. MOM maintains a repository of system events and can provide a knowledge base of operational procedures.

MOM provides two benefits that complement the other two management applications:

- Enterprise-level support, monitoring hundreds of computers and providing a centralized view of this information
- Increased resolution of your .NET-based application monitoring picture using management packs

MOM enables you to configure monitoring in many ways. As a rule of thumb, keep your monitoring environment as simple as you can while still getting the job done.

Note: Configuring MOM is a major subject in itself, so this chapter focuses only on those areas that are relevant to monitoring .NET-based applications — in particular, monitoring the FMStocks application within the context of the Fitch and Mather data center environment.

Operations Manager Components

MOM consists of components that can work in a distributed environment. The Windows 2000 core management services provide the main management data for Operations Manager. Additional providers include IIS, .NET Framework, COM+, Application Center, and AppMetrics.

Note: Even though the data management providers and core management services are not part of the MOM architecture, this discussion covers them to give you a complete picture of how these components come together.

The main MOM architectural components are as follows:

- Agents
- Consolidator Agent Manager
- Data Access Server
- Microsoft SQL Server Database

MOM Agents are intelligent monitoring components installed on each monitored Windows NT– or Windows 2000–based computer. Agents collect and analyze information and execute commands that MOM sends. They also store rules locally and can act without referring back to their managing computers.

The Consolidator Agent Manager (CAM) delivers rules and configuration data to the agents on the managed nodes. The CAM handles all communications with the managed computers on the network and sends information received from managed nodes to the Data Access Server (DAS). This information appears in the Microsoft SQL Server 2000 database. The CAM also automatically deploys and updates the remote agents on each managed computer and ensures that new rules propagate to the local agents.

MOM services use the DAS to access the database for reading or writing information. The DAS acts as a broker, transposing simple requests into database-specific tasks.

The Microsoft SQL Server database stores all event information and rules logic and is where the MOM management packs reside. Additionally, this database contains the prescriptive advice and Knowledge Base links, and the reporting engine queries it when generating reports.

Although you can install all these services on a single computer, spread them across multiple computers for better performance. The distributed nature of the MOM architecture makes it easy for you to avoid bottlenecks and support thousands of managed computers.

Management Packs

MOM management packs are a set of add-on modules and associated reports that facilitate monitoring and managing of Microsoft products. They also improve the availability, performance, and security of Microsoft Windows-based networks and server applications. With the addition of management packs, MOM centrally monitors and automatically resolves problems for networks of up to thousands of computers, continuously checking application software and servers.

MOM does not provide a management pack module that monitors serviced components. You therefore have two options:

- Monitor your COM+ components with Application Center 2000, and use the Application Center Management Pack to pass events and alerts to the MOM console.
- Use a third-party application, such as AppMetrics, to help you manage serviced components.

MOM management pack modules come preconfigured for use right after installation. Customize management pack modules by enabling and disabling rules or by adding additional rules.

MOM provides comprehensive event management for Windows 2000 Server, including Active Directory and IIS. The standard product also manages:

- Terminal Services
- Internet Information Server (IIS)
- .NET Framework
- Microsoft Distributed Transaction Coordinator (MDTC)
- Windows Internet Naming Services (WINS)
- Dynamic Host Configuration Protocol (DHCP)
- Domain Name Service (DNS)
- Routing and Remote Access Server (RRAS)
- Microsoft Transaction Service (MTS)
- Microsoft Message Queuing (MSMQ)
- Windows NT Version 4.0 System Event logs

Microsoft also supplies the Application Management Pack as an add-on to the standard management pack modules. These modules are:

- Exchange Server Versions 5.5 and 2000
- SQL Server Versions 7.0 and 2000
- Internet Security and Acceleration Server (ISA)
- Host Integration Server (HIS)
- Commerce Server

- Application Center
- Proxy Server Version 2.0
- Site Server Version 3.0
- SNA Server Version 4.0

Independent software and hardware vendors sell additional management packs to support Microsoft .NET Enterprise Servers and earlier Microsoft server applications.

The Fitch and Mather scenario in this module uses the following management pack modules:

- IIS Management Pack module
- .NET Framework Management Pack module
- Application Center 2000 Management Pack module
- SQL Server Management Pack module

Note: Microsoft Operations Manager Service Pack 1 is the basis for the management pack information in the following section.

IIS Management Pack Module

The IIS Management Pack module monitors the performance, availability, and security of Microsoft IIS versions 4.0 and later. This management pack module also monitors the Web server, FTP, SMTP, and NNTP interfaces. By detecting, notifying, and automatically responding to critical events, this management pack module helps to indicate, correct, and prevent IIS failures.

The IIS Management Pack module also alerts you to errors that customers receive as they browse your site. Additionally, it monitors events created by IIS and its attendant services in the Windows 2000 Event logs and the IIS logs.

This management pack module includes a script that polls and tracks the responsiveness of computers running IIS. This script provides the performance data to track usage trends and for capacity planning. The IIS Management Pack module also includes scripts to test and track Web page availability.

The IIS Management Pack module alerts you to the following critical conditions:

- Bad or missing links on your site or from a referring site
- Heavy Web traffic preventing users from connecting to your site
- Configuration errors or resource shortages affecting service levels
- Security issues such as attempts at a buffer overrun attack
- Active Server Page errors indicating a possible service outage of an application on your site

The IIS Management Pack module creates IIS-specific public views to give you a snapshot of the health of IIS.

.NET Framework Management Pack Module

The .NET Framework Management Pack is part of the management pack included with Microsoft Operations Manager Service Pack 1. This module delivers monitoring capabilities for the following core .NET Framework components:

- Common language runtime
- ASP.NET
- System Data
- Network Class Library

Application Center 2000 Management Pack Module

The Microsoft Application Center 2000 Management Pack module collects events generated by Application Center 2000 computers running on Windows 2000. This module ensures that your Web applications, and the clusters they belong to, work optimally and achieve the proper levels of availability. The Application Center 2000 Management Pack module also collects WMI events created by Application Center 2000. The module highlights events that might indicate possible service outages, configuration problems, and security issues so that you can quickly take corrective or preventative action. Some of the critical conditions this module alerts you to include:

- **Clustering**
  - Cluster controller unavailable
  - Cluster controller errors and failures
  - Cluster name resolution errors and failures
  - Cluster time synchronization errors and failures
- **Monitoring**
  - Event Logging errors and failures
  - Event Logging Agent errors and failures
  - Performance Logging and Counter errors and failures
- **Replication**
  - Replication Authentication errors and failures
  - Replication HTTP connection errors and failures
  - Object Replication errors and failures
  - Application Replication errors and failures
- **Request Forwarding**
  - Request Forwarding services unavailable
  - Request Forwarding thread pool errors and failures

- **Health Monitor**
  - Data collector status events that change to Critical or OK
  - System status events that change to Critical or Good

This management pack module monitors Application Center 2000 clusters centrally from the MOM console. It also provides the framework to consolidate some of the regular monitoring tasks associated with Application Center 2000.

The Application Center console and the Health Monitoring (HealthMON) console are the primary monitoring and management tools to keep Application Center clusters and applications healthy. However, the management pack can reduce the time and resources required to monitor Application Center 2000.

Note: Service Pack 1 for Microsoft Operations Manager also includes the Health Monitor console. It does not replace Health Monitor in Application Center 2000 but provides the ability to centralize and publish Health Monitor monitoring information through MOM.

SQL Server Management Pack Module

The Microsoft SQL Server module monitors computers running either SQL Server Version 7.0 or SQL Server 2000. The SQL Server module highlights events that may indicate service outages or configuration problems, so you can quickly take corrective or preventative actions. For example, this module alerts you to the following critical conditions:

- Deadlock problems
- Blocking issues
- Replication failure errors
- Log shipping related errors
- SQL Server unavailable
- SQL Server can no longer accept connections
- SQL Server process crashes
- SQL Server cannot allocate memory for normal operations

The SQL Server 2000 module supports monitoring of events, services, and performance counters for multiple instances of SQL Server 2000 on the same computer. This module also includes scripts to measure SQL Server availability.

Understanding the Architectural Design

The Fitch and Mather management solution architectural design centers on the basic design of the Microsoft core management services. Microsoft implements management services in the Windows platform so that applications can both read and write data to and from these management services. An application that writes data to any of the core management services is a *provider,* and an application that reads data from any of them is a *consumer.*

Therefore, the architectural design consists of the following three basic roles:

- Providers
- Core management services
- Consumers

Provider and consumer are not mutually exclusive roles. For example, Application Center 2000 and AppMetrics are both providers and consumers. On the other hand, IIS and ASP.NET act only as providers.

Figure 5.3 illustrates the architecture of the management solution.

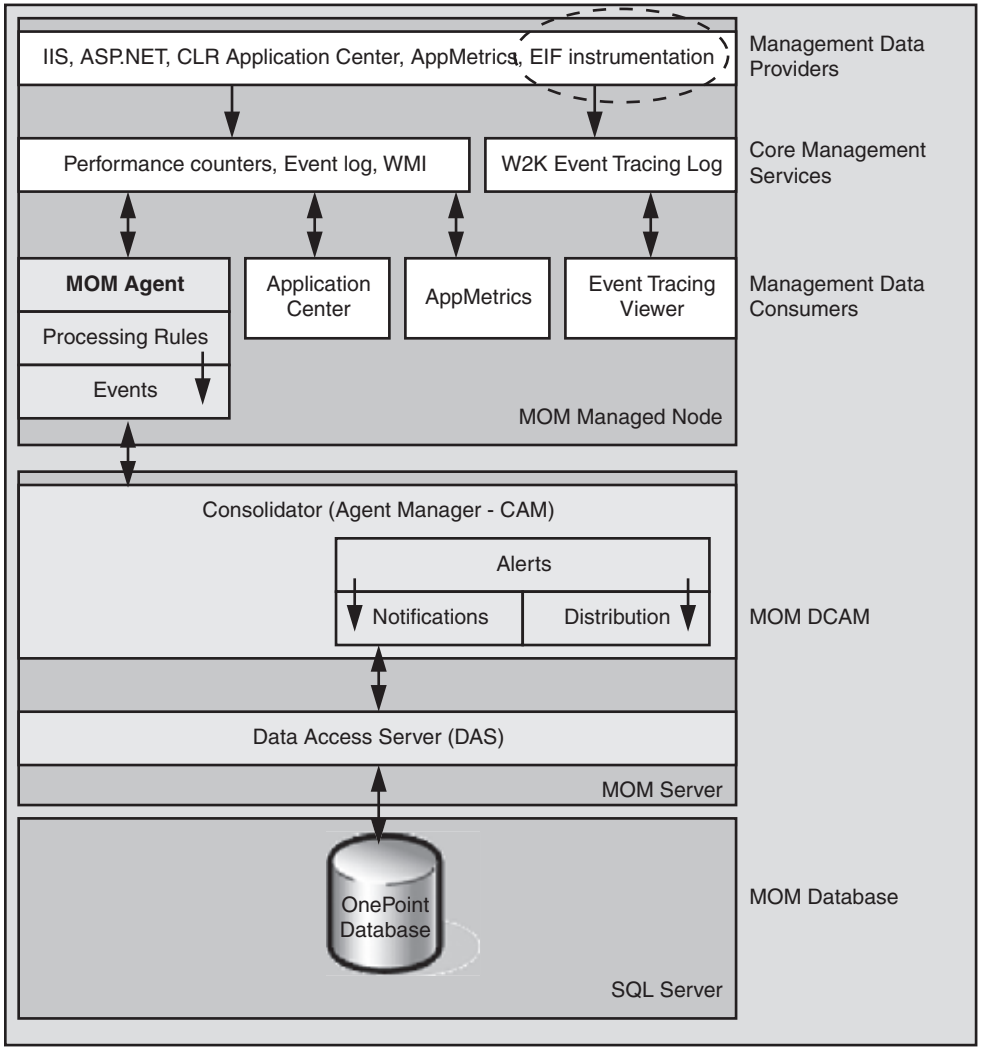


Figure 5.3
Fitch and Mather management architecture

Reviewing Core Management Services

The core management services in Windows 2000—performance counters, Event Log, WMI, and Event Tracing—have their own characteristics and area of application, requiring you to assess your technical and business requirements before deciding on which one to use. MOM simplifies this task using management packs that extend the capacity of MOM.

Performance Counters

Windows 2000 defines the performance data it collects in terms of objects, counters, and instances. A *performance object* is any measurable resource, application, or service.

Each object has several counters that measure performance, such as transfer rates for hard disks or, for processors, processor time consumed. Objects may also have one or more instances, which are unique occurrences of a particular object type. Not all object types provide multiple instances, however. An example of multiple instances is where you have individual processors on a multiprocessor computer.

The System Monitor console displays performance data in real-time charts or reports, and displays logged data from files. The Performance Logs and Alerts tool logs performance counter values over time and lets you configure alerts that can generate notifications should values exceed or fall below pre-set limits.

Event Logging

A core component of Windows 2000 is the Event Log service. The Event Log service starts automatically when you start Windows 2000 and records events in three main event logs:

- **Application log—**Lists events that applications running on the computer generate, including those application developers use to report application operations
- **Security log—**Tracks auditing events, such as successful and unsuccessful logons
- **System log—**Records events such as the success or failure of services starting or whether a computer shuts down unexpectedly

The Event Viewer Microsoft Management Console (MMC) lets you monitor and view system events recorded by the Event Log service.

Note: Depending on the computer role, you may see other logs, such as the Directory Services log or the DNS log.

Windows Management Instrumentation

Windows Management Instrumentation (WMI) is Microsoft's implementation of Web Based Enterprise Management (WBEM). WMI provides comprehensive management and monitoring facilities that you can access either locally or remotely.

WMI is independent of the operating system and simplifies the process of accessing management information programmatically.

Namespaces

The namespace is central to WMI. A namespace groups classes and instances together, controlling their scope and visibility. Namespaces are more like logical databases than physical locations and often contain specific classes and instances.

Typically, a namespace contains a set of classes and instances that represent managed objects in a particular environment. For example, the classes and instances defined to manage objects in the Microsoft Win32® API environment exist in their own unique namespace.

Each WMI installation contains a Root namespace, which only contains other namespaces. Under Root are the Default, Security, CIMv2, and any user-defined namespaces. Microsoft provides the Default namespace as a working area within WMI, and developers can add their own namespaces to this area. The Security namespace, under Root, contains the classes and instances used for the WMI security subsystem.

Using WBEMTEST Tool

You can use the Web Based Enterprise Management (WBEM) test tool called WBEMTEST to navigate the WMI namespace and discover what root namespace, classes, and instances your .NET-based application creates. WBEMTEST installs with WMI and is a default component of Windows 2000 and Windows XP.

Note: Use the WMI Control MMC snap-in to configure WMI properties, such as WMI Logging level, WMI log file size, and security.

The WBEMTEST tool is convenient and easy to use, but it is not the only tool available to navigate the WMI namespace. Alternatively, use some of the tools that come with the WMI SDK or download WMI administrative tools from the Microsoft.com Download Center at *http://www.microsoft.com/downloads/release.asp?ReleaseID=40804&area=search&ordinal=5*.

► To navigate the WMI namespace using the WBEMTEST tool

1. Log on to the target computer using an account that has administrative rights. If you are accessing the WMI namespace on a remote computer, be sure that the User ID you are logging on with has administrative rights on that computer.
2. Click **Start**, and then click **Run**.
3. Type in *WBEMTEST* and press **Enter**.
4. In the **Windows Management Instrumentation Tester** dialog box, click **Connect**.
5. In the **Connect** dialog box, if you are connecting to the local computer, click **Login**.

6. If you are connecting to a remote computer, in the **Namespace** box, type \\<*ServerName*>*root*, where ***ServerName*** is the name of the remote computer, and then click **Login**.
7. Click **Enum Classes**.
8. In the **Superclass Info** dialog box, select **Recursive** and click **OK**.

   You do not need to type anything in the text field.
9. In the **Query Result** dialog box, scroll down the list of top-level classes, and double-click **_NAMESPACE**.
11. In the **Object editor for _NAMESPACE** dialog box, click **Instances** to get a list of registered namespaces.
12. In the **Query Result** dialog box, to access your application's namespace, close all WBEM dialog boxes except **Windows Management Instrumentation Tester**.
13. In the **Windows Management Instrumentation Test** dialog box, click **Connect**, and type \\<*serverName*>*root*\<*NETAppNameSpace*> in the **Namespace** box. Then click **Login**.

Windows Event Tracing

You can use event tracing to define, record, and track activities, called *events*. An event is any discrete activity of interest, especially regarding performance. Event tracing records events in the order in which they occur. Use event tracing for capacity planning and detecting system bottlenecks.

Windows 2000 or later provides events that event tracing can log, such as disk I/O and page faults. Application developers can also define event types to use with event tracing. Examples of application events are the start and end of a transaction, such as a search operation.

Because event tracing minimally affects system performance, it can act as a low-impact debugger to troubleshoot timing-related problems in device drivers. For additional information about event tracing and device drivers, see the documentation that comes with the Windows Driver Development Kit (DDK).

Customizing the Management Applications

Now that you understand the management applications that can help you create your management architecture, you can now move on to look at how to customize these management applications. In this section, you will cover specific actions for monitoring.NET-based application.

Customizing Application Center

You can customize Application Center with a minimum investment of effort and time to provide a coarse view of IIS and your .NET-based application. MOM will assist you with creating a fine-grained picture of your .NET-based application.

You can take advantage of the Sample Monitors in Application Center to help add your initial monitors. To do so, add the monitors highlighted in Figure 5.4 to the Application Center Monitors and Web Site Monitors groups in the Synchronized Monitors group.

The overall procedure is as follows:

1. Copy the Sample Monitors.
2. Configure the monitors.
3. Assign an e-mail action.
4. Customize the Localhost Monitor.

For detailed procedures, see later sections in this chapter.

Figure 5.4
Adding the highlighted monitors to the Synchronized Monitors group

Note: You can carry out similar procedures to monitor your Business tier.

Copy the Sample Monitors

Before proceeding, be sure you are working at the cluster controller; otherwise, the next cluster synchronization will remove the new monitors.

► **To copy the Sample Monitors**

1. Click **Start**, point to **Programs**, point to **Administrative Tools**, and then click **Application Center**.
2. In the left pane, expand **Health Monitor**.
3. If your cluster controller is not in the list of monitor computers, right-click **All Monitored Computers**, and then click **Connect to another computer**.
4. Select a computer from the list, and then click **OK**.
5. Expand <*computername*>.
6. Expand **Synchronized Monitors**, and expand **Application Center Monitors**.
7. In **Synchronized Monitors**, drag the following monitors onto the **Application Center Monitors** group:
   - **SELECT * FROM MicrosoftAC_Cluster_Loadbalancing_Event**
   - **SELECT * FROM MicrosoftAC_Cluster_Controller_Discovery_Failed_Event**
8. Expand **Cluster (Load Balancing) Events**.
9. In **Synchronized Monitors**, drag the following monitors to the **Application Center**:
   - **SELECT * FROM MicrosoftAC_Cluster_Membership_Server_Live_Event**
   - **SELECT * FROM MicrosoftAC_Cluster_Membership_ServerFailed_Event**
   - **SELECT * FROM MicrosoftAC_Cluster_Loadbalancing_ServerOffline_Event**
10. In **Sample Monitor**, expand **Web Site Monitors**.
11. In **Synchronized Monitors**, drag the following monitors individually from **Application Log** onto the **Web Site Monitors** group:
    - **W3Svc errors and warnings**
    - **WebService**

Configure the Monitors

Your next step is to configure the monitors.

► **To configure the monitors**

1. In **Health Monitor**, expand **Synchronized Monitors (Application Center)**.
2. Expand **Application Center Monitors**.
3. Right-click **Request Forwarding Failure**, click **All Tasks**, and select the **Disabled** check box.

You disable this monitor because in this configuration, you are not using Request Forwarding.

4. Enable the following monitors by right-clicking each of them:
   - **SELECT * FROM MicrosoftAC_Cluster_Loadbalancing_Event**
   - **SELECT * FROM MicrosoftAC_Cluster_Loadbalancing_ServerOffline_Event**
   - **SELECT * FROM MicrosoftAC_Cluster_Membership_Server_Live_Event**
   - **SELECT * FROM MicrosoftAC_Cluster_Membership_ServerFailed_Event**
   - **SELECT * FROM MicrosoftAC_Cluster_Controller_Discovery_Failed_Event**
   - **Synchronization Session Errors**
5. Point to **All Tasks**, and clear the **Disabled** check box.

Once you have configured the monitors, you can enable them.

► **To enable the Web Sites Monitors**

1. In **Health Monitor**, expand **Synchronized Monitors (Application Center)**.
2. Expand **Web Sites Monitors**.
3. Enable the following monitors by right-clicking each of them:
   - **http://127.0.0.1/**

     Note: Enable the http://127.0.0.1 monitor only on the Presentation tier or on defined Web sites. If the Business tier runs only COM components, this monitor is not beneficial.

   - **W3Svc errors and warnings in Application Log**
   - **WebService**
4. Click **All Tasks**, and clear the **Disabled** check box.

Assign an E-mail Action

Assign the **Email Administrator** action to the **Application Center Monitors** group, and set this action to occur whenever a monitor in the group goes into a critical state.

► **To assign an e-mail action**

1. Right-click the **Application Center Monitors** group, and select **Properties**.
2. Click the **Actions** tab.
3. Click **New Action Association**.
4. Select **Email Administrator**, and then click **OK**.
5. Perform steps 1 through 4 on the **Web Site Monitors** group.

Customize the Localhost Monitor

Exercise common sense when setting up this monitor because the Microsoft Operations Manager IIS Management Pack module provides similar functionality. You usually do not want to generate identical events and alerts reporting the same problem. However, you may have an operations team focusing on .NET application management and using Application Center rather than on the MOM console. In this case, you may want to set both this monitor and the one provided with MOM.

► **To customize the Localhost Monitor**

1. Right-click the **http://127.0.0.1** monitor, and then select **Properties**.
2. On the **General** tab, change the path name to correspond to the path for the .NET-based application—in this case, **http://127.0.0.1/fmstocks7**.
3. Click the **Details** tab, and then enter the local URL for the .NET-based application, which for the FMStocks example is **http://127.0.0.1/fmstocks7**.
4. Click the **Actions** tab, and then click the **New Action Association** button. In the **Execute Action Properties** dialog box, select the **Log to Websitefailures.log** check box.
5. Click **OK**.
6. In the **Thresholds** pane, right-click the **HTTP Monitor – ResponseTime > 30 seconds** threshold, and then select **Properties**.
7. On the **General** tab, change the monitor name to *HTTP Monitor – ResponseTime > 3 seconds*.
8. Click the **Expression** tab, and in the **Is greater than** box, enter ***3000***. Click **OK**. This value is in milliseconds.

Using the Application Center 2000 Resource Kit

The tools in the Application Center 2000 Resource Kit let you customize Application Center in many ways. This resource kit lets you create a multitude of monitors and actions in addition to configuring performance counters as you need them.

Three tools in the resource kit are particularly useful when monitoring .NET-based applications:

- AppMetricsLite
- Advanced Configuration Editor
- Cluster Monitor

The AppMetricsLite tool gives you an appreciation for what AppMetrics for Transactions can do. AppMetricsLite works only when the user interface is running, and it does not deploy agents.

The Advanced Configuration Editor gives you greater control over Application Center, covering numerous advanced settings that do not appear in the Application

Center console. This editor lets you configure both local and remote Application Center clusters.

Cluster Monitor enables you to monitor Application Center clusters remotely. You can monitor from any computer running either Application Center Server or the Application Center administrative client.

For more information about these tools, see Appendix F to the Application Center Resource Kit on the Microsoft Web site at *http://www.microsoft.com/technet/prodtechnol/acs/reskit/acrkappf.asp*.

In the next section, you look at how you can integrate a third-party utility into your monitoring framework. This lets you collect monitoring information on individual application components.

Customizing AppMetrics

To monitor .NET-based applications with AppMetrics, you must install an AppMetrics Agent on each of the computers in your Business tier and install a Manager computer to process the component metrics. Separating the collection from the analysis and storage minimizes the additional software that you must install on your Business tier and reduces the overhead of collecting this fine-grained information. For more information about installing and configuring the AppMetrics Manager and Agent computers, see the AppMetrics documentation.

AppMetrics has two types of monitors: production and diagnostics. For health monitoring of your serviced components, you want to create a Production Monitor.

Using Production Monitors

The Production Monitor collects data for one or more COM+ Application processes and gathers this information for each component with a class. The Production Monitor reduces monitoring overhead on the application server by aggregating the statistics for all component instances by class.

The aggregate values arise from measuring individual component instances and collecting metrics over a specified time interval. The default time interval is 1 minute, but most production applications benefit from using a 5-minute collection interval. Using a 5-minute interval greatly reduces the size of the collected data and smoothes out short-term usage peaks.

Create a single production monitor for each .NET-based application you want to monitor. This records the component information for each application along with all of the other measurement data you collect, but it lets you separate data from different applications as necessary.

The following instructions show you how to create a Production Monitor and configure an alert that tells you if you have component instances within the

FMStocks application that are not responding. You do this by setting a threshold value for component instance duration.

The actual value for component instance duration that you set depends on your components. Ideally, you should run AppMetrics for several weeks to collect usage statistics for your components before you set alert values. The reports available with AppMetrics provide you with the information that you need to set the proper alert values.

► **To create a Production Monitor and configure an alert**

1. On the AppMetrics Manager computer, click **Start**, point to **Programs**, point to **AppMetrics for Transactions**, and then click **AppMetrics for Transactions 2.5**.
2. Right-click **Application Monitors**, click **New**, and then click **Application Monitor**.

   The **New Application Monitor** dialog box appears.
3. In the **New Application Monitor** dialog box, click **COM+ Production Manager**, and then click **OK**.
4. Enter the name of your production monitor (in this example, it is *FMStock7 Production*) and a **Description**. Click **Finish**.
5. A new node appears as a child of the **Application Monitors** node with the name of your Production Monitor.
6. Expand the **Production Monitor** node (in this example, called **FMStocks7 Production**)—the child of the Application Monitor node.
7. Right-click the **Agents** node, click **New**, and then click **Agent** to bring up the **Add Agent** dialog box.
8. Enter the name of the computer where your components run (in the FMStocks example, "*NETAPP001*"), and then click **Browse** to bring you to the **Browse Agents** dialog box.
9. This dialog box shows you the agents that you can select. At this point, there aren't any, so click **New**. The **Create New Agent** dialog box appears.
10. In **Create New Agent**, enter the name of the agent for this monitor (in this example, *FMStocks7ProdAgent*). Click **OK** to create the agent. You return to **Browse Agents** with the new agent selected. Click **OK** to continue.
11. In **Add Agent**, click **OK**.

To customize this monitor, set up an alert.

► **To set up an alert**

1. Expand the **FMStocks7 Production** node, and then click **Detail**.
2. Under **Transaction Detail Level**, select **Identify by** and **Component**.

Figure 5.5 shows how your AppMetrics console should look.

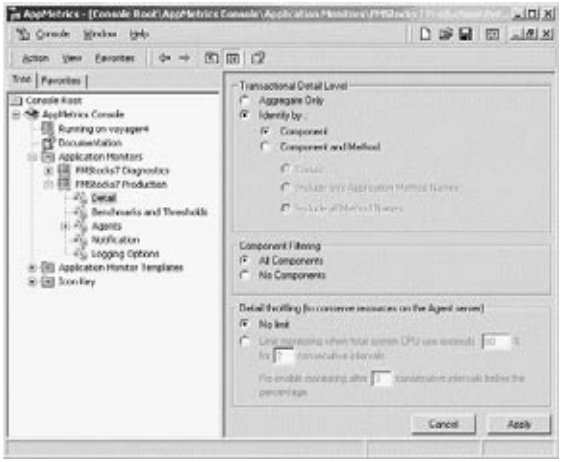


Figure 5.5
Production Monitor configuration

3. Click **Apply**.
4. Expand the **Agents** node, and then expand the agent that you want to monitor. Select the **Applications** node.
5. Click the option for **Select Application from list**, find the COM+ applications your application uses, and then select them. (For FMStocks, these are **FMStocks7.DAL** and **FMStocks7.GAM**).
6. Click **Apply**.

   Figure 5.6 shows how the selections for FMStocks should appear in the AppMetrics console.

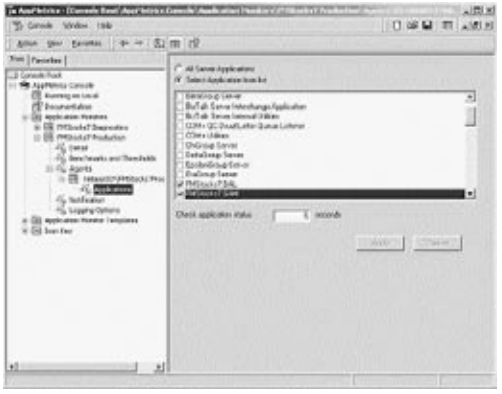


Figure 5.6
Production Monitor Agent configuration

7. Click **Logging Options**, select the option for **Automatic Rotation**, and under **Rotation Frequency**, select **Hourly**.

   Figure 5.7 shows the logging options, including options for uploading the database.

Figure 5.7
Setting up logging options

8. Click **Apply**. Your monitor is now ready to collect data.
9. To turn on the monitor, right-click the **FMStocks7 Production** node, and then click **Start Monitor**.

When you select a COM+ Library Application (such as FMStocks.DAL), AppMetrics marks the components in the COM+ Application for monitoring. Changes to the COM+ Application take effect when the COM+ Application first loads. When using AppMetrics with a .NET-based application, ASP.NET keeps Library COM+ Applications active until the ASP.NET worker process (Aspnet_wp.exe) recycles. For AppMetrics to monitor an ASP.NET-based application correctly, you must wait for the ASP.NET worker process to recycle, or you must restart it manually.

In the preceding steps, the FMStocks7 Production Monitor automatically rotates every hour. The FMStocks7 Production Monitor must be on for more than one hour before data uploads from the log files to the SQL Server database. Once the data is in the SQL Server database, analyze it using AppMetrics Excel reports.

AppMetrics can correlate the relationships between components. This functionality helps diagnose problems and tune performance. The AppMetrics Drill-Down Report analyzes the method calls between components. The AppMetrics Drill-Down Report gets its data from an AppMetrics Diagnostics Monitor. The Diagnostics Monitor collects significantly more data than an AppMetrics Production Monitor, so only run the diagnostics monitor for short periods.

Using Diagnostics Monitors

Create a Diagnostics Monitor using the same procedure as for creating a Production Monitor, except choose a different monitor template type and name.

► To create a Diagnostics Monitor

1. On the AppMetrics Manager computer, click **Start**, point to **Programs**, point to **Xtremesoft**, point to **AppMetrics for Transactions**, and then click **AppMetrics for Transactions 2.5**.
2. Right-click **Application Monitors**, click **New**, and then click **Application Monitor**. The **New Application Monitor** dialog box appears.
3. In **New Application Monitor**, click **COM+ Diagnostics Manager**, and then click **OK**.
4. Enter the name of your production monitor (in this example, *FMStock7 Diagnostics*) and a **Description**. Then click **Finish**.

   A new node appears as a child of the **Application Monitors** node with the name of your Diagnostics Monitor.
5. Expand the **Diagnostics Monitor** node (in our example, *FMStocks7 Diagnostics*)—the child of the **Application Monitors** node.
6. Right-click the **Agents** node, select **New**, and then click **Agent** to bring up the **Add Agent** dialog box.
7. Enter the name of the computer on which your components run (in the FMStocks example, *NETAPP001*), and then click **Browse** to bring you to the **Browse Agents** dialog box.
8. In the **Browse Agents** dialog box, click **New**.
9. In **Create New Agent**, enter the name of the Agent for this monitor (in this example, *FMStocks7DiagAgent*). Click **OK** to create the agent. When you return to the **Browse Agents** dialog box with the new agent selected, click **OK**.
10. Click **OK** to return to the **Add Agent** dialog box, and then click **OK** again.

As you did with the Production Monitor, you must configure the Diagnostics Monitor Agent to collect information from the FMStocks COM+ Applications.

► To set the COM+ Applications

1. Expand the **Agents** node, and then select the **COM+ Applications** node.
2. Click **Select Application from list**, and then select the COM+ applications that your application uses (for FMStocks, **FMStocks7.DAL** and **FMStocks7.GAM**). Click **Apply** to save your changes.
3. Click **Logging Options**, and then select the option for **Automatic Rotation**. Under **Rotation Frequency**, select **Hourly**.
4. Click **Apply** to save your changes.

The setup of your Diagnostics Monitor is complete. You can now collect data by starting and stopping the Diagnostics Monitor. Diagnostics Monitor generates more processor load on the system it monitors and creates large amounts of data on the manager, so you should run Diagnostic Monitors only for short periods.

To view the diagnostics information you collect about the relationships between components, use the AppMetrics Drill-Down Report. For information about running the Drill-Down Report, see Chapter 6, "Notifying and Reporting."

Customizing Web Monitor

You can use Web Monitor to generate external stimulus for your .NET-based application, and in this section, you configure Web Monitor to generate external stimulus for the FMStocks sample application. You can install Web Monitor on the Operations Manager computer for simplicity. Alternatively, install Web Monitor on a separate computer that connects to your .NET-based application from the clients' perspective (for example, from the Internet).

Note: Install the MOM agent on the monitored computer to collect all the events and alerts centrally for your .NET-based application.

Web Monitor is a set of JavaScript and XML files. Configure Web Monitor by editing these files with a simple text editor, such as Notepad.

► To install and configure Web Monitor

1. Download Web Monitor from the MSDN Web site at *http://msdn.microsoft.com/msdnmag/issues/0700/code/Kougiouris0700.exe*.
2. Double-click *Kougiouris0700.exe*. Install Web Monitor into its own directory by running the self-extracting archive. The default installation directory is C:\WebMon.
3. Edit Main.js and remove the failure recovery code by adding double slashes ("//") at the beginning of the following lines. Note that some of the lines are truncated to fit on this page. Changes are in bold.

```
//   if (!ret) { // At least one site PING failed.
//           strSubject = RestartServices()

//           var eMailFromNode = xmlRoot.selectSingleNode(…)
//           var eMailToNodeList = xmlRoot.selectNodes(…);
             // And let somebody know.
//           for (i = 0; i < eMailToNodeList.length; i++) {
//                   SendMessage(eMailFromNode.text,…);
//           }
//   }
```

4. Edit the *Urlstomonitor.xml* file to write EventLog records on failures. The **LOG** tag controls this behavior. The bold text shows this entry in the XML file.

```
<m:MONITOR xmlns:m="x-schema:monitorSchema.xml">
    <m:LOG LogSuccess="false" LogFailure="true"/>
```

5. Remove the e-mail sender and recipient addresses, as Web Monitor does not send e-mail. Leave the **E-MAIL** tags, as the schema definition requires their presence. The example code shows the **E-MAIL** section in bold.

```
<m:MONITOR xmlns:m="x-schema:monitorSchema.xml">
    <m:LOG LogSuccess="false" LogFailure="true"/>
    <m:E-MAIL>
    </m:E-MAIL>
```

6. Add a **SITE** tag for each computer in the Web cluster. A **SITE** tag consists of a pair of lines (a URL and **MATCH** tag). The Fitch and Mather architecture has three Web servers (*NETIIS001*, *NETIIS002*, *NETIIS003*).

   To validate the presence of the Fitch and Mather home page, configure Web Monitor to search for the text "Fitch and Mather". The following code illustrates a single entry. The bold text must match the text returned from the Web page.

```
<m:SITE NAME="netiis001">
    <m:URL>http://netiis001/fmstocks7</m:URL>
    <m:MATCH><![CDATA[Fitch and Mather]]></m:MATCH>
</m:SITE>
```

7. Add each SITE to the **SITES** tag. The bold text below shows the **SITES** tag for all three Web computers in Fitch and Mather. In this example, WebMon looks for the key words "technical difficulties" because that text appears on the Fitch and Matter customized page for HTTP errors. In your production environment, you would customize Web Monitor to suit your application and the error messages returned.

```
<m:MONITOR xmlns:m="x-schema:monitorSchema.xml">
    <m:LOG LogSuccess="false" LogFailure="true"/>
    <m:E-MAIL>
    </m:E-MAIL>
    <m:SITES>
        <m:SITE NAME="netiis001">
            <m:URL>http://netiis001/fmstocks7</m:URL>
            <m:MATCH><![CDATA[technical difficulties]]></m:MATCH>
        </m:SITE>
        <m:SITE NAME="netiis002">
            <m:URL>http://netiis002/fmstocks7</m:URL>
            <m:MATCH><![CDATA[technical difficulties]]></m:MATCH>
        </m:SITE>
        <m:SITE NAME="netiis003">
```

```
            <m:URL>http://netiis003/fmstocks7</m:URL>
            <m:MATCH><![CDATA[technical difficulties]]></m:MATCH>
        </m:SITE>
    </m:SITES>
</m:MONITOR>
```

8. Configure Task Scheduler to run Web Monitor every hour. On the Operations Manager computer, open Control Panel and launch **Scheduled Tasks**. Right- click the **Scheduled Task** window, select **New**, and then click **Scheduled Task**.
9. If a dialog box appears saying that the service is currently stopped, click **Yes**, and then click **Start**, point to **Programs**, point to **Administrative Tools**, and point to **Services.** Locate the **Task Scheduler** service, and in the Task Scheduler Properties dialog box, configure the service for automatic startup.
10. Rename the task **New Task** to **Web Monitor**. The task name will already be highlighted and in edit mode.
11. Right-click the **Web Monitor** task, click **Properties**, and then click the **Task** tab.
12. Enter the command line to run into *one line* in the dialog box. This example assumes that you installed Windows 2000 into C:\winnt.

    ```
    c:\winnt\system32\wscript.exe c:\webmon\testmon.wsf c:\webmon\urlstomonitor.xml
    ```

13. Click the **Schedule** tab, and then select **Daily**.
14. Enter the **Start time** for the Web Monitor task, and then click **Advanced** to set the **Duration** and **Timing**.
15. Click **OK** to close the **Properties** dialog box.
16. Click **OK** on **Advanced Schedule Options**.
17. Click **OK** on Web Monitor. A **Run as** dialog box appears.
18. Enter the account information with the appropriate rights in the dialog box, and click then **OK**.

Customizing Microsoft Operations Manager

In this book, the assumption is that your .NET-based application is of critical importance. For that reason, you want to view all related events and alerts generated by this application independently from other applications in the enterprise. MOM lets you do this in a repeatable and reliable fashion.

One method of doing this is to customize the processing rule groups (PRG). However, you should weigh the benefits of this action against the resultant management overhead. When you copy a rule from one PRG to another, it becomes a new rule. Updating the original rule—for example, by installing a service pack—does not update the copied rule. Therefore, you must manually track down any copied rules and update them.

An alternative approach is customizing Public Views in MOM and generating a view for only your .NET-based application. In your own environment, you can create more elaborate and sophisticated configurations to monitor your systems and applications and to meet your technical and business requirements.

Note: To carry out the actions in the next section, install Microsoft Operations Manager Service Pack One. To do this, follow the instructions in Appendix B of the MOM Installation Guide, "Upgrading Operations Manager."

The sequence of actions is as follows:

1. Create computer groups
2. Customize public views
3. Monitor IIS
4. Monitor the .NET Framework
5. Monitor serviced components
6. Monitor Data tier
7. Monitor instrumentation-generated events
8. Monitor synthetic transaction events

Creating Computer Groups

To organize the computers running your .NET-based application, create three Computer Groups (CG).

► **To create Computer Groups**

1. Log onto the MOM computer, click **Start**, point to **Programs**, point to **Microsoft Operations Manager**, and click **MOM Administrator Console**.
2. In the left pane, expand **Microsoft Operations Manager** (**Default**). Expand **Rules**, and then select **Computer Groups**.
3. On the **Action** menu, click **Create Computer Group**.
4. In the **Computer Group Properties – Computer Types** dialog box, click **Next**.
   The default is all computer types selected.
5. In **Computer Group Properties – Computers**, select **Only computers whose name matches**.
6. In the **Domain name** drop-down list box, select **equals**. In the **Domain name** box, type the name of the domain.
7. In the **Computer name** drop-down list box, select **matches wild card**. In the **Computer name** box, enter the common letters for the Presentation tier computer names, followed by an asterisk (for example **netiis***). Click **Next**.

8. In **Computer Group Properties – Formula**, click **Next**. A **Microsoft Operations Manager** message box appears with the text **The Computer Group formula field is empty. This will cause the rule to match ALL computers. Do you want to continue?** Click **OK**.
9. In **Computer Group Properties – Excluded Computers**, click **Next**.
10. In **Computer Group Properties – Included Computers**, click **Next**.
11. In **Computer Group Properties – General**, in the **Name** box, enter *FMStocks Presentation Tier Servers – CG*. Then click **Finish**.
12. Click **No** if a dialog box appears with the text **Would you like to deploy a group of processing rules to the computers matching this newly created computer group?**
13. Repeat the above steps to create the *FMStocks Business Tier Servers – CG* and the *FMStocks Data Tier Servers – CG* Computer Groups.

    Enter the proper wildcard characters in Step 7 and the proper group name in Step 11.

Customizing Public Views

To customize the view of your .NET-base application servers, create a Public View.

► **To create a Public View**

1. Log onto the MOM computer, click **Start**, point to **Programs**, point to **Microsoft Operations Manager**, and click **MOM Administrator Console**.
2. In the left pane Expand **Microsoft Operations Manager** (**Default**), and expand **Monitor**. Select **Public Views**.
3. On the **Action** menu, click **New**, and then click **Computer View**.
4. In **Computer View Properties – View Type**, select **Computers that satisfy specified criteria**, and click **Next**.
5. In **Computer View Properties – Criteria**, select **in specified computer group**, and click **running Microsoft Operations Manager agent**.
6. On the **View description** pane, click **specified**.
7. Enter the computer group name, using wildcard characters if necessary. (In the Fitch and Mather example, enter *fmstocks**.) Click **OK**. Click **Next**.
8. In the **Computer View Properties - General** dialog box, enter *FMStocks Servers* in the **View Name** field.
9. Click **Finish**.

Note: In Step 5, include the criteria "running Microsoft Operations Manager Agent" because in this example, Web Monitor runs on the monitored computer. Ideally, you would install Web Monitor on another computer and include that computer in the Public View.

Monitoring IIS

You can increase the fine-grained information from IIS by using the Microsoft Operations Manager IIS Management Pack. This approach lets you centralize IIS, NLB, and Application Center monitoring on the MOM console using the Application Center 2000 Management Pack module.

Management Pack modules are usually ready to use and do not require additional configuration. However, using the IIS Management Pack module requires the IIS settings on the managed computers to meet certain requirements:

- Configure IIS log files, as the IIS Management Pack module monitors these log files.

  For information about how to configure logging on IIS, see "HOW TO: Enable Logging in IIS 5.0" on the Microsoft Product Support Services Web site at *http://support.microsoft.com/default.aspx?scid=KB;en-us;313437&.*

- Monitor no more than five IIS sites (virtual directories) for each MOM client.

  If a MOM agent applies rules that use the IIS application log provider, the agent reads the logs from the first five sites. If more logs exist on the agent computer, the agent generates an alert.

- Let only 200 files maximum accumulate in IIS log file directories. This reduces the load on the CPU.

- Limit the logging generated by the IIS server.

The MOM agent reads the log files using a low-priority thread, so with large IIS log files and a busy computer, the MOM agent might not process the log file quickly. In exceptional circumstances, the MOM agent can keep the log file open for more than 24 hours. Decrease the amount of time required to process a log file by limiting the number of logged events. However, if the agent holds a log file open for an unreasonable amount of time, close the log file by restarting the OnePoint service.

Figure 5.8 illustrates the Processing Rule Groups that the IIS Management Pack module creates.

In Figure 5.8, you can see that the IIS Management Pack module comes with predefined rules for IIS version 4.0 and IIS version 5.0. The bulk of the rules exist in both folders under the IIS Shared Rules; however, they are independent of each other. You can disable rules in **IIS 4.0 – IIS Shared Rules**, but keep them enabled under **IIS 5.0 – IIS Shared Rules**.

Some of the IIS counters defined in Chapter 3, "Selecting Data Sources" for are either not monitored by the MOM management packs by default or need updating. To resolve this problem, this next example shows you how to create a Computer Rule Group to group all additional IIS counters. You then create a rule for the Current Connections performance counter.

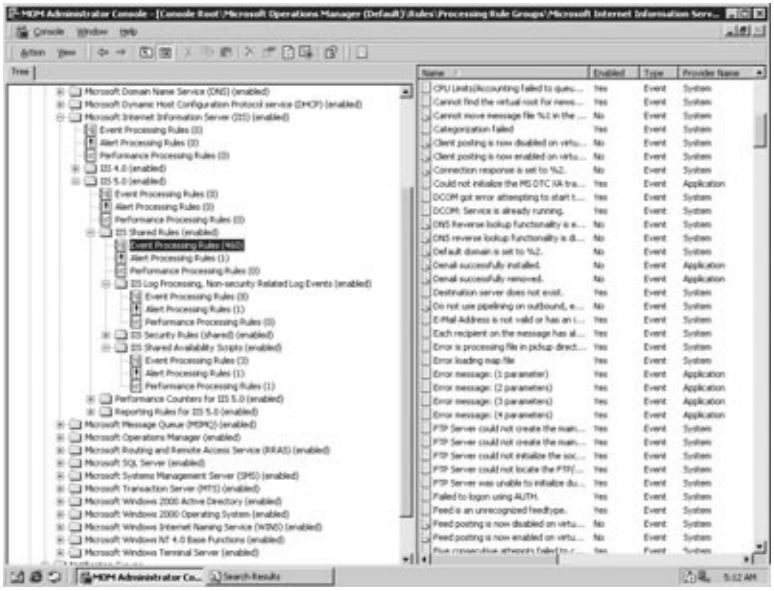


Figure 5.8
IIS processing rule groups

Creating the Computer Rule Group is not mandatory, but it helps to keep the additional counters together. Remember that if you create new rules in the IIS Management Pack module PRG, you risk losing them if you update the IIS module and select the replace option.

► **To create a rule to monitor the Current Connections counter**

1. Log on to the MOM computer, click **Start**, point to **Programs**, point to **Microsoft Operations Manager**, and click **MOM Administrator Console**.
2. In the left pane, expand **Microsoft Operations Manager** (**Default**), expand **Rules**, and then expand **Processing Rule Groups**.
3. Select **Processing Rule Groups,** and then on the **Action** menu, click **Create Processing Rule Group**.
4. Enter *IIS Rules – PRG* in the **Name** field, and then click **Next**.
5. Click **Finish**.
6. At the question **Would you like to deploy the processing rules in this newly created processing rule group to a group of computers?**, click **Yes**.
7. In **Processing Rule Group Properties**, click the **Computer Groups** tab, and then click **Add**.
8. Select **FMStocks Presentation Tier Servers – CG** and click **OK**. Click **Add** again and select **FMStocks Business Tier Servers – CG**, and click **OK** twice.

9. Expand the newly created Processing Rule Group **IIS Rules – PRG** and select **Performance Processing Rules**.
10. On the **Action** menu, click **Sample Performance Data**.
11. At the **Performance Measure Processing Rule Properties – Data Provider** window, scroll down in the **Provider Name** drop-down list box and select **Web Service–Current Connections–_Total–15–minutes**, and then click **Next**.
12. At the **Performance Measure Processing Rule Properties – Schedule** window, click **Next**.
13. At the **Performance Measure Processing Rule Properties – Responses** window, click **Next**.
14. At the **Performance Measure Processing Rule Properties – Knowledge Base** window, click **Next**.
15. At the **Performance Measure Processing Rule Properties – General** window type *Web Service – Current Connections,* and click **Finish**.
16. In the left pane, right-click **Rules**, and select **Commit Configuration Change**.

The Current Connections rule currently measures performance. If you know the maximum current connections your application can handle, you can create a threshold processing rule. This lets you specify a threshold, which will generate an alert once the counter value exceeds the threshold.

In the next steps, you create a Computer Rule Group to group all additional system counters. You then create a rule for the Available Megabytes performance counter. As with the Current Connections counter in the previous example, creating the Computer Rule Group is not mandatory.

► **To create a rule to monitor the Available Megabytes counter**

1. Log on to the MOM computer, click **Start**, point to **Programs**, point to **Microsoft Operations Manager**, and click **MOM Administrator Console**.
2. In the left pane, expand **Microsoft Operations Manager** (**Default**), expand **Rules**, and then expand **Processing Rule Groups**.
3. Select **Processing Rule Groups**, and then on the **Action** menu, click **Create Processing Rule Group**.
4. Enter *System Rules – PRG* in the **Name** field, and then click **Next**.
5. Click **Finish**.
6. At the question **Do you want to deploy this PRG to a group of computers**, click **Yes**.
7. In **Processing Rule Group Properties**, click the **Computer Groups** tab, click **Add**.
8. Select **FMStocks Presentation Tier Servers – CG** and click **OK**. Click the **Add** button, select **FMStocks Business Tier Servers – CG**, and click **OK** twice.
9. Expand the newly created Processing Rule Group **System Rules – PRG**.

10. On the **Action** menu, click **Compare Performance Data**.
11. At the **Threshold Processing Rule Properties – Data Provider** window, scroll down in the **Provider Name** drop-down list box and select **Memory – Available Mbytes – 15–minutes**, and then click **Next**.
12. At the **Threshold Processing Rule Properties – Schedule** window, click **Next**.
13. At the **Threshold Processing Rule Properties – Criteria** window, click **Next**.
14. At the **Threshold Processing Rule Properties – Threshold** window, select **the sampled value**. Under **Match when the Threshold value becomes**, select **less than** and enter a value of *10*, then click **Next**.
15. At the **Threshold Processing Rule Properties – Alert** window, select **Generate Alert,** select **Warning** in the **Alert severity** drop-down list box, and then click **Next**.
16. At the **Threshold Processing Rule Properties – Alert Suppression** window, click **Next** (enabled by default).
17. At the **Threshold Processing Rule Properties – Responses** window, click **Next**.
18. At the **Threshold Processing Rule Properties – Knowledge Base** window, click **Next**.
19. At the **Threshold Processing Rule Properties – General** window, type *Memory – Available Mbytes* in the **Name** field, and then click **Finish**.
20. In the left pane, right-click **Rules**, and select **Commit Configuration Change**.

The Microsoft Windows 2000 Operating System Management Pack module has predefined rules to monitor the Processor counters. Follow these steps to update the threshold information for these counters:

► **To update the Processor counter rules**

1. Log on to the MOM computer, click **Start**, point to **Programs**, point to **Microsoft Operations Manager**, and click **MOM Administrator Console**.
2. In the left pane, expand **Microsoft Operations Manager** (**Default**), expand **Rules**, expand **Processing Rule Groups**, expand **Microsoft Windows 2000 Operating System**, expand **Windows 2000 – All Computers**, and expand **Threshold Performance counters for Windows 2000**.
3. Select **Performance Processing Rules**, and double-click the **% Total Processor Time (95) averaged over 6 samples** rule. The **Threshold Processing Rule Properties** dialog box appears.
4. Update the **Name** field to **% Total Processor Time (75) averaged over 6 samples**.
5. Click the **Threshold** tab, change the value of **greater than** from **95** to **75**, and then click **OK**.

Note: The **Threshold Performance counters** group in the Windows 2000 Management Pack module also has the counters defined for **% Privilege time**, **% User Time** and **% Interrupt time.** Be careful when updating the Windows 2000 Management Pack module not to replace these counters, which will remove any customized settings that you define.

The next step is to enable and configure the HTTP Ping rule. These steps show you the procedure for the FMStocks sample application, but you can amend these to suit your requirements.

► **To enable and configure the HTTP Ping rule**

1. Log onto the MOM computer, click **Start**, point to **Programs**, point to **Microsoft Operations Manager**, and click **MOM Administrator Console**.
2. In the left pane, expand **Microsoft Operations Manager** (**Default**), **Rules**, **Processing Rule Groups**, and then **Web Monitor Synthetic Transactions – PRG**.
3. Right-click **Event Processing Rules**, click **New**, and then click **Event Processing Rule**.
4. In **Select Event Processing Rule Type**, select **Alert on or Respond to Event**, and click **Next**.
5. In **Event Processing Rule Properties – Data Provider**, in the **Provider Name** box, select **Schedule every 2 minutes**, and click **Next**.
6. In **Event Processing Rule Properties – Schedule**, click **Next**.
7. In **Event Processing Rule Properties – Responses**, click **Add**, and select **launch a script**.
8. On the **Script name** list, click **HTTP Ping**, and then click **Edit**.
9. In **Script Properties**, click the **Script** tab.
10. Look for the line **If Trim(strURL) = "" Then strURL = "http://127.0.0.1"**, and change it to **If Trim(strURL) = "" Then strURL = "http://127.0.0.1/fmstocks7"**.)
11. Click **OK** twice.
12. Click **Next** twice.
13. In **Event Processing Rule Properties – General**, enter *FMStocks Servers*.
14. Click **Finish**.

Note: In Step 9, you can edit the HTTP script and create your own rules. Disable any rules that you do not need.

Figure 5.9 shows the Application Center Processing Rule Groups created by the Application Center 2000 Management Pack module.

The Application Center 2000 Management Pack not only lets you create a centralized monitoring view but also allows you to monitor Application Center itself. This feature is especially useful in situations in which Application Center experiences problems that it cannot report.

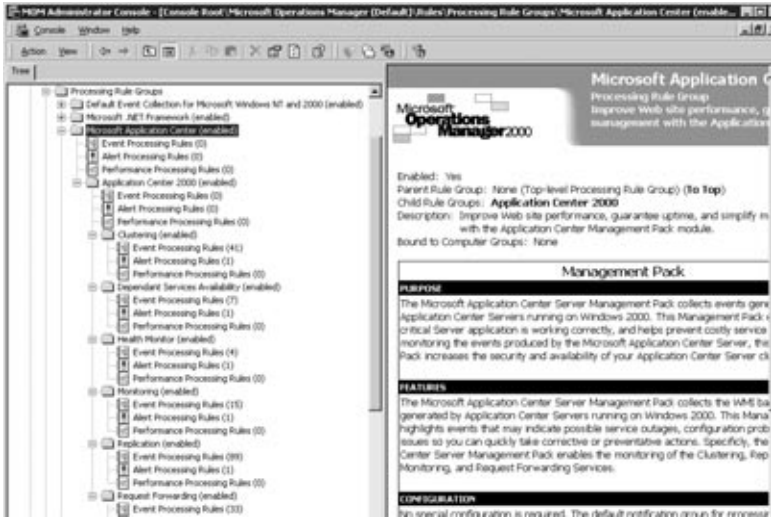


Figure 5.9
Application Center 2000 Management Pack Processing Rule Groups

Monitoring the .NET Framework

The .NET Framework Management Pack module monitors the .NET Framework technologies ASP.NET, common language runtime, and .NET Remoting. Microsoft Operations Manager Service Pack 1 includes this module.

Figure 5.10 illustrates the processing rule groups created by the .NET Framework Management Pack module.

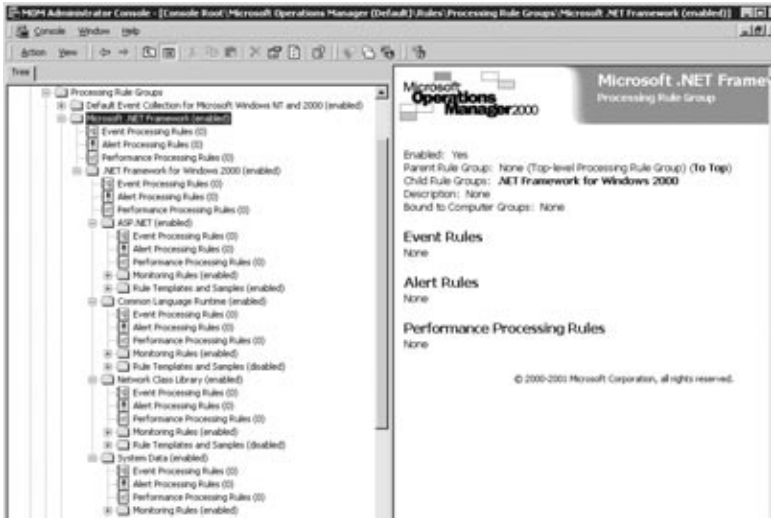


Figure 5.10
.NET Framework Management Pack Processing Rule Groups

In Figure 5.10, you can see that the .NET Framework Management Pack module creates four child processing rule groups:

- **ASP.NET**
- **Common Language Runtime**
- **Network Class Library**
- **System Data**

The .NET Framework Management Pack module captures performance counters and events as part of monitoring the different logical subdivisions (ASP.NET, common language runtime, .NET Remoting). Chapter 3, "Selecting Data Sources," describes these performance counters and events.

Note: The common language runtime monitors .NET Remoting. The performance monitor object is **.NET CLR Remoting**.

You will need to create or update some rules for the .NET Framework. The next steps guide you through the creation of Microsoft Operations Manager rules to monitor the .NET Framework counters and to enable any initially disabled rules.

► To enable ASP.NET rules

1. Log on to the MOM computer, click **Start**, point to **Programs**, point to **Microsoft Operations Manager**, and click **MOM Administrator Console**.
2. In the left pane, expand **Microsoft Operations Manager** (**Default**), expand **Rules**, expand **Processing Rule Groups**, expand **Microsoft .NET Framework**, expand **.NET Framework 1.0**, expand **ASP.NET**, and then expand **Monitoring Rules.**
3. Select **Performance Processing Rules**.
4. Enable the following rules by double-clicking them, selecting the **Enabled** box, and clicking **OK**.
   - Performance Measuring ASP.NET\Request Wait Time
   - Performance Measuring ASP.NET\Request Rejected
   - Performance Measuring ASP.NET\Request Queued

These steps create the additional rules for the ASP.NET counters.

► To create ASP.NET performance counter rules

1. Log on to the MOM computer, click **Start**, point to **Programs**, point to **Microsoft Operations Manager**, and click **MOM Administrator Console**.
2. In the left pane, expand **Microsoft Operations Manager** (**Default**), expand **Rules**, and then expand **Processing Rule Groups**.
3. Select **Processing Rule Groups**, and then on the **Action** menu, click **Create Processing Rule Group**.

4. Enter **ASP.NET Rules – PRG**, and click **Next**.
5. In the **Processing Rule Group Properties** window, click **Finish**.
6. At the question **Would you like to deploy the processing rules in this newly created processing rule group to a group of computers?**, click **Yes**.
7. In **Processing Rule Group Properties**, click the **Computer Groups** tab, and then click **Add**.
8. Select **FMStocks Presentation Tier Servers – CG**, and click **OK** twice.
9. Expand the newly created Processing Rule Group **ASP.NET Rules – PRG**. Select **Performance Processing Rules**.
10. On the **Action** menu, click **Sample Performance Data**.
11. At the **Performance Measure Processing Rule Properties – Data Provider** window, click the **New** button. Select **Windows NT Performance Counter**, and then click **OK**.
12. At **Windows NT Performance Counter Provider Properties – General** window click the **Browse** button. Select **Remote Computer**, enter the name of a Presentation tier server (*NETIIS001* in the FMStocks7 sample application), and click **OK**.
13. Click the **Object** field, select **ASP.NET Applications** from the drop-down list box. Click the **Counter** field and select **Pipeline Instance Count**. Select the **Instance** field and select your <*.NET Application instance*>. Click **Next**.
14. At **Windows NT Performance Counter Provider Properties – Synchronization** window, click **Finish**.
15. At the **Performance Measure Processing Rule Properties – Data Provider** window, click **Next**.
16. At the **Performance Measure Processing Rule Properties – Schedule** window, click **Next**.
17. At the **Performance Measure Processing Rule Properties – Responses** window, click **Next**.
18. At the **Performance Measure Processing Rule Properties – Knowledge Base** window, click **Next**.
19. At the **Performance Measure Processing Rule Properties – General** window, type *ASP.NET Application – Pipeline Instance Count*, and then click **Finish**.
20. In the left pane, right-click **Rules**, and select **Commit Configuration Change**.

Repeat steps 9 to 20 and create rules for the **Cache Total Entries** and **Cache Total hit ratio** counters.

► **To enable common language runtime rules**

1. Log on to the MOM computer, click **Start**, point to **Programs**, point to **Microsoft Operations Manager**, and click **MOM Administrator Console**.
2. In the left pane, expand **Microsoft Operations Manager** (**Default**), expand **Rules**, expand **Processing Rule Groups**, expand **Microsoft .NET Framework**, expand **.NET Framework 1.0**, expand **Common Language Runtime**, and then expand **Monitoring Rules**.
3. Select **Performance Processing Rules**.
4. Enable the following rules by double-clicking them, selecting the **Enabled** box, and then clicking **OK**.
   - Performance Measuring .NET CLR Exceptions(_Global_)\# of Exceptions Thrown / sec
   - Performance Measuring .NET CLR Memory(_Global_)\# Bytes in all Heaps
   - Performance Measuring .NET CLR Remoting(_Global_)\Context-Bound Objects Alloc/Sec
   - Performance Measuring .NET CLR Remoting(_Global_)\Remote Calls/Sec

Monitoring Serviced Components

AppMetrics provides a rich monitoring environment for serviced components, whether these serviced components run as managed or unmanaged code. To provide a centralized monitoring view of your .NET-based application, integrate AppMetrics with MOM.

In AppMetrics, you can set a variety of alert conditions for both process-level metrics and component-level metrics, as described in Chapter 3, "Selecting Data Sources." Configure the notification mechanism in AppMetrics to write to the Windows Event Log. Then enable MOM to watch for these events in the event logs of your computers running these serviced components. The section "Customizing AppMetrics," earlier in this chapter describes how to do this.

► **To set alerts from AppMetrics**

1. On the AppMetrics Manager computer, click **Start**, point to **Programs**, point to **Xtremesoft**, point to **AppMetrics for Transactions**, and then click **AppMetrics for Transactions 2.5**.
2. In the right-hand pane, expand **Application Monitors**. Then expand **AppMetrics Production Monitor (FMStocks7 Production)**.
3. Select **Benchmarks and Thresholds**.
4. Click the **Components** tab. Select the relevant **Application** and **Component**, and then set the **Current Benchmark** value for the component's **Duration**.
5. To set up a notification, click **Notification**.

6. In the **Notification Configuration** pane, click **Add**, select **Windows Event Log**, and set **Priority** to **High**.
7. In **Add delivery mechanism**, click **OK**.
8. In the **Notification Configuration** pane (shown in Figure 5.11), click **Apply**.

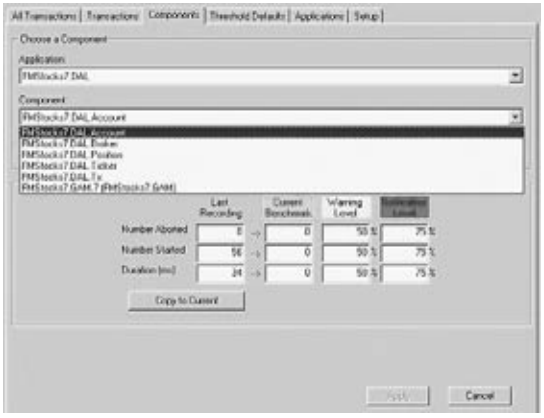


Figure 5.11
AppMetrics Notification setup

Note: AppMetrics organizes component information based on the COM+ Application process that activates the component. For COM+ Library applications such as FMStocks7.DAL and FMStocks7.GAM, the GAM components appear under the FMStocks7.DAL Application in the **Benchmarks and Thresholds** node. Figure 5.12 shows an example of this.

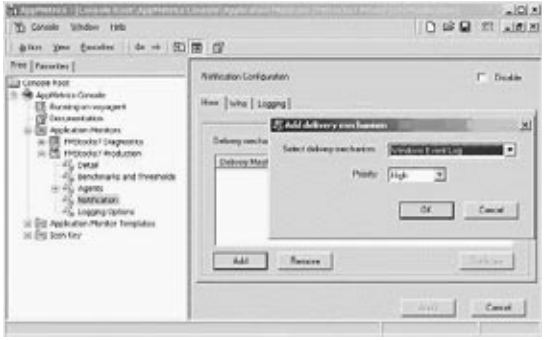


Figure 5.12
Setting Library application thresholds

After you configure AppMetrics to log events in the Windows Event Log, configure MOM to watch for these events.

► **To create a Microsoft Operations Manager rule to monitor for the events generated by AppMetrics**

1. Log on to the MOM computer, click **Start**, point to **Programs**, point to **Microsoft Operations Manager**, and then click **MOM Administrator Console**.
2. In the left pane, expand **Microsoft Operations Manager** (**Default**), expand **Rules**, and then expand **Processing Rule Groups**.
3. On the **Action** menu, click **Create Processing Rule Group**.
4. Enter **AppMetrics Events – PRG**, and click **Next**.
5. Click **Finish**.
6. At the question **Would you like to deploy the processing rules in this newly created processing rule group to a group of computers?**, click **Yes**.
7. In **Processing Rule Group Properties**, click the **Computer Groups** tab, click **Add**.
8. Select **FMStocks Business Tier Servers – CG**, and then click **OK** three times.
9. Select the newly created Processing Rule Group **AppMetrics Events – PRG**.
10. On the **Action** menu, click **Alert on or Respond to Event (Event)**.
11. In the **Provider** name list, click **Application**, and then click **Next**.
12. Select the **with event id** check box, and then enter **4** in the **event id** box. Click **Next**.
13. In **Schedule**, select **Always process data**, and then click **Next**.
14. Select **Generate alert**. Click **Next**.
15. In **Alert Suppression**, clear the **Suppress duplicate alerts** check box, and then click **Next**.
16. Click **Next** two more times.
17. In **General**, in the **Name** box, enter *Collecting AppMetrics Events – PRG*, and then click **Finish**.
18. In the left pane, right-click **Rules**, and select **Commit Configuration Change**.

Monitoring the Data Tier

To monitor the Data tier, use the SQL Server Management Pack module. Figure 5.13 illustrates the Processing Rule Groups that the SQL Server Management Pack module creates with the event processing rules in the right-hand pane.

Use the SQL Server Management Pack module with Microsoft Operations Manager Service Pack 1, but note that installing this service pack disables several default MOM rules. However, these disabled rules are low priority or only useful in certain circumstances.

The SQL Server Management Pack module captures the Performance Monitors discussed in Chapter 3, "Selecting Data Sources." It can also monitor for SQL Server service events recorded in the event viewer log.

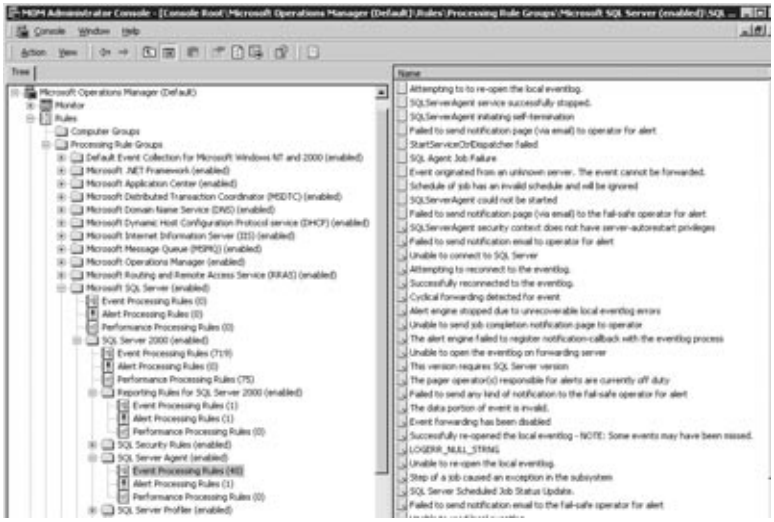


Figure 5.13
SQL Server Management Pack Processing Rule Groups

Monitoring Synthetic Transaction Events

To monitor synthetic transaction events, configure Web Monitor to generate an entry in the Windows 2000 Event Log should the Web site home page be unavailable. Then create rules to look for these failure events. To do this, create a Processing Rule Group for Web Monitor, and then create rules to watch for Web Monitor events.

► To create a Processing Rule Group for Web Monitor

1. Log onto the MOM computer, click **Start**, point to **Programs**, point to **Microsoft Operations Manager**, and click **MOM Administrator Console**.
2. In the left pane, expand **Microsoft Operations Manager** (**Default**), expand **Rules**, and then expand **Processing Rule Groups**. Then click **Processing Rule Groups**.
3. On the **Action** menu, select **Create Processing Rule Group**.
4. Enter *Web Monitor Synthetic Transactions – PRG*, and click **Next**.
5. Click **Finish**.
6. At the question **Would you like to deploy the processing rules in this newly created processing rule group to a group of computers?**, click **Yes**.
7. On the **Computer Groups** tab in **Processing Rule Group Properties**, click **Add**.
8. Select the **FMStocks Presentation Tier Servers – CG**, and click **OK** twice.

Now that you have created a processing rule group, create rules for that group that look for Web monitor failures.

► **To create rules to watch for Web Monitor events**

1. Expand the newly created **PRG Web Monitor Synthetic Transactions – PRG**.
2. Right-click **Event Processing Rules**, click **New**, and then click **Event Processing Rule**.
3. In **Select Event Processing Rule Type**, select **Alert on or Respond to Event**, and click **Next**.
4. In **Event Processing Rule Properties – Data Provider**, in the **Provider Name** box, click **Application**, and click **Next**.
5. In **Event Processing Rule Properties – Criteria**, check the **from source** box, enter *WSH*, check the **with event id** box, and enter *1*. Click **Next**.
6. In **Event Processing Rule Properties – Schedule**, select **Always process data**, and click **Next**.
7. In **Event Processing Rule Properties – Alert**, check the box **Generate alert**, and click **Next**.
8. In **Event Processing Rule Properties – Alert Suppression**, click **Next**.

   The default setting enables alert suppression.
9. Click **Next** twice.
10. In **Event Processing Rule Properties – General**, in the **Name** box, type *Web Monitor Failure*.
11. Click **Finish**.
12. In the left pane, right-click **Rules**, and select **Commit Configuration Change**.

Summary

In this chapter, you looked at detailed procedures for setting up your management applications and consolidating monitoring information into MOM. In addition, you reviewed how to increase the resolution of your .NET-based application monitoring picture using MOM management packs.

You saw how the MOM management pack modules make the task of setting up a management infrastructure considerably easier. Finally, you worked through configuring MOM to monitor the FMStocks application and reviewed the monitoring support provided by AppMetrics.

References

For more information on Microsoft Operations Manager, go to the Microsoft Operations Manager Web site at *http://www.microsoft.com/mom/*.

For prescriptive guidance on deploying and configuring Microsoft Operations Manager in the enterprise, see "Microsoft Systems Architecture: Internet Data Center" on the Microsoft TechNet Web site at *http://www.microsoft.com/technet/itsolutions/idc/default.asp.*

For greater detail on Application Center 2000, go to "Cluster It and Scale It Out" on the Microsoft Web site at *http://www.microsoft.com/applicationcenter/.*

For more information about AppMetrics for Transactions from Xtremesoft Inc, go to the Xtremesoft Web site at *http://www.xtremesoft.com/solutions/trans_product_tour.htm.*

6

Notifying and Reporting

Introduction

So far you have looked at methods of setting up monitoring and alerting. These methods allow you to be sure that an alert will be generated or an event will be recorded if a particular incident occurs or a service level is transgressed.

An increasing number of enterprises run business-critical IT services at availability levels of 99.9 percent or better. Acting swiftly and correctly can make the difference between achieving acceptable levels of service uptime and failing to do so. You need to create a framework that will handle a range of possible occurrences using monitoring, checking, and alerting combined with automatic and manual responses. You need to do this in a manner that is economically justified in relation to the value of the information infrastructure that you are monitoring and the severity of the reported faults.

You must define notification levels and actions within your operating procedures. This means that if you receive an urgent e-mail at your desk from your monitoring infrastructure, or your pager is quietly vibrating in your waistband (while you were hoping to enjoy a quiet night at the cinema), you and your operations team will know what to do instantly.

The key is planning and foresight. Without a roadmap of documented operational procedures and detailed actions (such as actions on service failure), you will find that when an event occurs, you are unable to react correctly to resolve the issue. Without prior planning, it will be impossible for you to devise and implement effective automatic problem-solving routines.

Reporting allows you to turn the raw monitoring data into easily assimilated information about the health of your system. This information then becomes the basis for implementing proactive maintenance on your .NET-based applications. Reporting

can also provide high-level management information in accordance with the principles of the Microsoft Operations Framework (MOF).

In this chapter, you cover converting these alerts and events into notifications, as well as the actions you should take once you receive such a notification. You look at the different notification levels, how to escalate notifications, and how to ensure notification reliability. Finally, you cover creating reports using management applications.

Notifications and Actions

Consider this scenario: it is a wet Saturday afternoon and the World Series final (substitute football, cricket, golf, or tiddly-winks as appropriate) is starting. The teams come onto the field and you look forward to a serious afternoon's baseball. Just as you settle back into your recliner, you hear a strident beeping from your cell phone. On the LCD screen, the following message appears:

"1457–Application failure: Application process FMStocks.DAL terminated abnormally 1 times in the last 5 minutes."

Great. So now what? Do you:

1. Ignore it. You're not missing this game for nothing or nobody!
2. Run out of the house, jump in your car, and go screeching off to your data center, cutting off everyone on the freeway. Once you arrive there, heap abuse on your operations team for not sorting the problem out themselves.
3. Fire up your PC, connect to your data center, and reboot the server running the application. Because let's face it, you don't really trust your duty technician to do it right. And rebooting a computer always solves the problem, right?
4. Call your duty operations team and check that they are aware of the situation and that they have everything under control. Tell them you are available if they need you and then get back to watching the game.
5. Ignore it for now. The automatic monitoring will restart the application. If it fails again, you'll check with your duty technician.
6. What cell phone?

When responding to events generated by your .NET-based applications, consider the level of notification and subsequent response that is appropriate to the situation. In the case above, the termination of FMStocks.DAL must have been reasonably serious (otherwise you as the operations manager would not have been alerted), but probably not so serious that you need to escalate the readiness state of your data center. Without an appreciation of what FMStocks.DAL actually does and what the consequences of it failing are, it is impossible to say what the correct action to take is.

But what if FMStocks.DAL should fail three more times in the next few minutes? The situation has now become more serious and your relaxing afternoon in front of the telly could be in jeopardy. You must define clear guidelines that encompass the severity of an alert, precisely defined service-failure categories, and properly documented responses. Without these, it is difficult to ensure that your operations team carry out appropriate actions according to the situation.

Defining Appropriate Actions

Four absolutes govern appropriate actions. Your operational responses need to be:

- Effective
- Timely
- Economically justified
- Repeatable

Effective

An effective response rectifies the problem as quickly as possible without compounding the situation. This might involve a single action (either manual or automatic) that immediately solves the issue. In a more complex case, this would involve a sequence of troubleshooting procedures, which may need evaluating before you continue.

Effectiveness varies according to a range of factors. For example, stopping and starting a service once might solve a problem, but if it doesn't, repeating that action is unlikely to produce any further benefits. However, if you stop a service, reinstall the .NET component on which that service depends, and restart the service, you may resolve the issue.

Detailed troubleshooting steps in your operating procedures increase the effectiveness of any remedial actions. Any member of your operations team must be able to apply the troubleshooting steps, not just the personnel who are intimately familiar with .NET-based applications.

Timely

A timely response is one that occurs within an acceptable period after you receive notification of a problem. The time of this response will link to your service level agreement (SLA) and contracted availability level.

With a non-essential component, it might be quite acceptable to wait until Monday morning, whereas a more urgent alert might justify getting half your support team out of bed at 3:00 A.M. on New Year's Day. This is assuming, of course, that they have gone to bed by then. Hopefully, the last example won't happen too often.

Economically Justified

Monitoring alert responses must be justified in terms of the cost of the response and the expense of maintaining the infrastructure to generate that response. When considering these costs, include both direct costs arising from service unavailability and indirect costs from loss of confidence, clients moving to competitors, and so on.

Repeatable

Responses need to be repeatable in that your operations team must apply fixes consistently. One major issue with service management is loss of critical troubleshooting information that results from failing to document successful problem resolution.

You need to avoid troubleshooting each service outage from scratch. Implement management practices to ensure that your operations team learns from previous experience.

Note: Microsoft Operations Manager (MOM) provides an excellent framework for gathering, recording, and accessing information resulting from service issues.

Defining Failure Levels

When discussing appropriate actions in relation to service failures, consider the following factors:

- What levels of overall alert severity do you need to define?
- What levels of service failure do you need to specify?
- What is the relationship between alert severity and operational effects?
- What levels of operational readiness do you need to define?
- How will a certain alert level affect your infrastructure?
- How urgently do you need to respond to each alert type?
- How do you prevent your operations team from being overwhelmed by alerts?
- Is it appropriate to restore services immediately?

Alert Severity and Relative Importance

Your monitoring and alerting infrastructure provides information on service availability issues that arise from your .NET-based applications. However, not all alerts are equal. Classify alert severity to prioritize dealing with a particular issue.

Typical operational procedures might define the following four levels of alert severity:

- Informational
- Priority
- Urgent
- Immediate

Informational

Informational alerts do not affect operational effectiveness or SLA levels. An example of an informational alert for a .NET-based application is when the **Memory: Private Bytes** value for the application exceeds a preset limit.

Priority

Priority alerts cover events that may affect network operations in the future; they need to be rectified within 60 minutes. An example of a priority alert for a .NET-based application is where the application response time exceeds a certain value, such as 500 milliseconds.

Urgent

Urgent alerts include most events that affect SLAs and network operations. Urgent alerts require that your staff works on the problem as soon as possible and, during quiet hours, might involve calling in additional personnel. In a .NET-based application environment, application failure on one cluster member triggers an Urgent alert.

Immediate

Immediate alerts are caused by a total service failure. If the monitoring environment generates an immediate alert, every member of the technical staff must carry out their prescribed actions to get the service operational again in the shortest time possible. However, as described in the section "Restoring Services" later in this chapter, getting the service running again must be consistent with troubleshooting the reason that the service failed.

An Immediate alert occurs when synthetic end-to-end transactions fail to complete. Hopefully, your monitoring environment will detect this level of catastrophic failure before your end-users start ringing you up to complain.

Service Failure Categories

Alert severity levels need to link to particular service failure categories. Service failure categories are significantly more specific than alert severity levels because they relate to a particular issue.

Typical service failure categories include:

- Performance degradation
- Severe performance degradation
- Predictive failure
- Imminent predictive failure
- Network connectivity outage
- Computer down

- Cluster down
- Hardware component failure
- Mains power supply failure
- External network attack (for example, distributed denial of service)
- Intrusion attempt
- Security breach

The categorization ensures that your operations team has a clear understanding of the threats to service availability and the severity of each threat. This increases staff responsiveness by enabling them to diagnose the alert severity level accurately.

Defining Readiness States

Your action plan must define readiness states in which you can immediately communicate your current level of preparedness. Readiness states must be logical, understandable, and linked to the situation's severity. An example system might employ the following four states:

- Normal
- Elevated
- High
- Highest

Normal

During normal operations, the organization maintains everyday staffing levels and alertness. Members of the operations team can take holidays or attend courses. A duty technician covers the nighttime shift and he or she responds to any alerts.

Elevated

If the duty technician receives a priority alert, then the readiness state rises to Elevated. This means that a second technician goes to standby and the network manager is notified of the change of the readiness state.

Elevated alert states apply in times of predicted high-service demand. This might include the time right before national holidays or, in the case of the Fitch and Mather Web site, a time that the stock market rises or falls by more than 50 points in a day. During extended periods of elevated alert, this might lead to the recall of all staff at non-essential training courses.

High

The High readiness state applies when an urgent alert is received, during times of actual or predicted excessive-service demand, or if the network is under attack. Operations team members must return from training courses and cannot depart on holiday during this time. A team of two technicians occupies the data center, and a duty manager is on call.

Highest

This readiness state applies during times of national emergency or if an Immediate alert is received that is not resolved within 15 minutes. Three technicians and a duty manager occupy the data center at all times and all staff must return, including those on holiday.

Changing Alert States

Your operating procedures should detail the service failures and events that trigger a change in readiness state. These procedures should also specify who in the managerial hierarchy can authorize changing from a higher to a lower readiness state.

Preventing Alert Overload

Alert overload arises when the number and frequency of alerts or notifications overwhelms your operations team. Alert overload can affect your team in two ways:

- Excessive alert levels impede spotting critical information.
- High alert levels cause performance degradation, thus exacerbating the problem.

To prevent alert overload, set monitoring thresholds correctly and ignore repeated alerts of the same type. MOM ignores repeated alerts if you use the **Suppress duplicate alerts** setting on the **Alert Suppression** tab. This lets you suppress duplicate alerts that have the same **Alert Description**, **severity**, **source name**, and so on. For information on how to configure alert suppression, see the section on "Monitoring Serviced Components" in Chapter 5, "Configuring Management Applications."

Filter alerts through tools such as Event Viewer to remove noise caused by excessive alert generation. However, this is a short-term or emergency measure. Once you have resolved the immediate crisis, take steps to prevent further alert overload, such as increasing the suppression of duplicate alerts.

Restoring Services

Restoring a .NET-based application into service as quickly as possible is not always the right thing to do. Restarting the service could destroy evidence about what caused the failure or, as in the case of a security breach, allow the attack to continue. If your SLA does not require immediate service resumption, analyze the situation and work out why the service failed in the first place.

Implementing Notifications

Once you have defined your alert severity levels, service failure categories, and readiness states, specify who will receive notifications under particular circumstances. When defining whom to notify, follow these general guidelines:

- Create a notification hierarchy
- Use notification methods

- Combine notification methods
- Understand notification reliability

Creating a Notification Hierarchy

When you define a notification hierarchy, your operations team must understand who needs to be notified for each level of alert severity. MOM uses notification groups that map to specific operational responsibilities. However, you may have different requirements and already have notification procedures in place.

Prevent plaguing your managers with notifications by referring only important alerts up the notification hierarchy. Use alert severity levels and service failure levels to ensure that alerts of a particular severity go to the right people. This provides effective notification without crying "Wolf!" unnecessarily.

Using Notification Methods

Your choice of notification method is extremely important in terms of the immediacy and reliability of the notification transport method. Table 6.1 shows the main notification methods.

Table 6.1: Comparison of notification methods

Method	Advantages	Disadvantages	Comments
E-mail	Works across distributed networks Allows for large amounts of information in message Quick	May be subject to delays outside your control Requires recipient to be logged on to e-mail	Best for more general monitoring and low priority notifications Useful as a backup to more immediate methods Good during working hours Combined with a wireless e-mail device, could provide alternative to mobile phone or pager
Pager	Very quick Discreet Convenient	Recipient may be out of range Limited message length Pager needs to be on	Best for immediate notifications
Screen pop-ups	Very quick More immediate than e-mail	Recipient needs to be using a computer on the network Requires NetBIOS name resolution	Best for high-priority alerts Must be used in combination with other methods

Method	Advantages	Disadvantages	Comments
Short Messaging Service (SMS) on mobile phone	Quick Allows longer messages than a pager	Recipient may be out of range Limited message length Phone needs to be on	Now a viable alternative to a pager
Audible alarms	Instant	Short range Can be distracting Do not convey much information unless combined with speech messages	Good for immediate notifications in the data center Could be supplemented with a klaxon/alarm outside the data center
Visible alarms	Instant	Do not convey much information unless on display screens Someone has to be looking at them	Best when combined with audible alarms, but beware of the "Three Mile Island" effect where multiple alarms cause sensory overload

Combining Notification Methods

Combine notification methods to ensure that the message gets through. For urgent of notifications, use all available options, including the big arm that reaches out of the server rack and taps your operators on the shoulder.

Table 6.2 shows methods of escalating notifications, and links alert levels to notification types. More serious alerts trigger more immediate and pervasive notification methods.

Table 6.2: Notification method escalation

Alert Level	Notification Type
Informational	E-mail
Priority	Screen pop-ups, e-mail
Urgent	Pager, screen pop-ups, e-mail
Immediate	Audible and visible alarms, screen pop-ups, pager, e-mail

Understanding Notification Reliability

Notification reliability covers delivery mechanism reliability and the dependability of the method that dispatches notifications to the delivery mechanism. Investigate the complete delivery path, from your monitoring solution to the recipient.

For example, when using MOM to deliver e-mail notifications, look at how MOM communicates with the e-mail computer. MOM lets you configure e-mail connectivity either through Simple Mail Transfer Protocol (SMTP) or through Microsoft Exchange Server using Messaging Application Programming Interface (MAPI). However, you can only specify one Exchange Server or SMTP computer to transmit e-mail notifications. For maximum reliability, use either a clustered Exchange 2000 virtual server or a load-balanced SMTP array as the e-mail notification computer.

Testing Notifications

Test your notification methods on a regular basis to check that they work. Ensure that your team members respond correctly and within the required time frame. This testing should be a component of your operational procedures.

Notification Example

This example shows how to configure MOM to send a notification when your .NET-based application is not operational because of overloading. Although this might seem straightforward, it is not always easy to accomplish this reliably. MOM is good at identifying fatal errors, such as Internet Information Services (IIS) failures, ASP.NET restarting, and so on, but it may not identify intermittent or performance-related problems.

The Fitch and Mather data center has configured Application Center and MOM based on the steps in "Creating Computer Groups" and "Monitoring Synthetic Transaction Events" in Chapter 5, "Configuring Management Applications." Two workstations run Application Center Test to generate the loading on the Web site. The machine.config file limits system resources.

Figure 6.1 illustrates the FMStocks Computer View. In the details pane, you can see all the computers running FMStocks and the computer running MOM.

Note: The MOM computer appears in this graphic is because it runs Web Monitor.

Figure 6.1 shows two of the application servers reporting errors, one of them with at least one critical error. The four Web servers indicate errors and the MOM computer shows a critical error reported by Web Monitor.

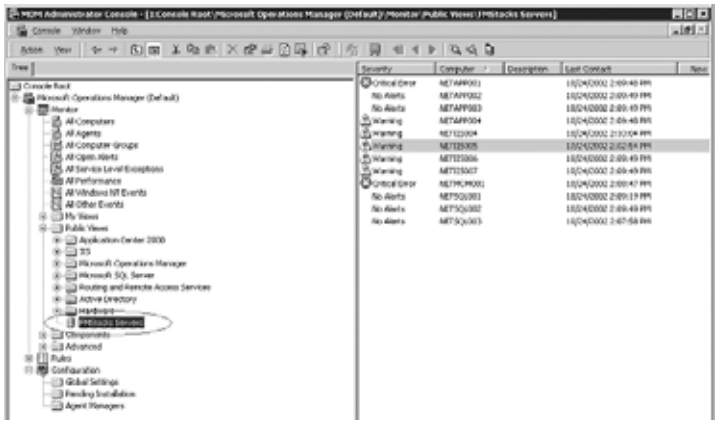


Figure 6.1
FMStocks Computer View

In Figure 6.2, you can see the following errors reported by the local computer (NETIIS004):

- Two Windows Management Instrumentation (WMI) critical errors from the Application Center 2000 management pack module
- Two warning errors from the HTTP Ping event rule in the IIS management pack module
- One warning error from a Web Site Unavailable message reported by the IIS management pack module

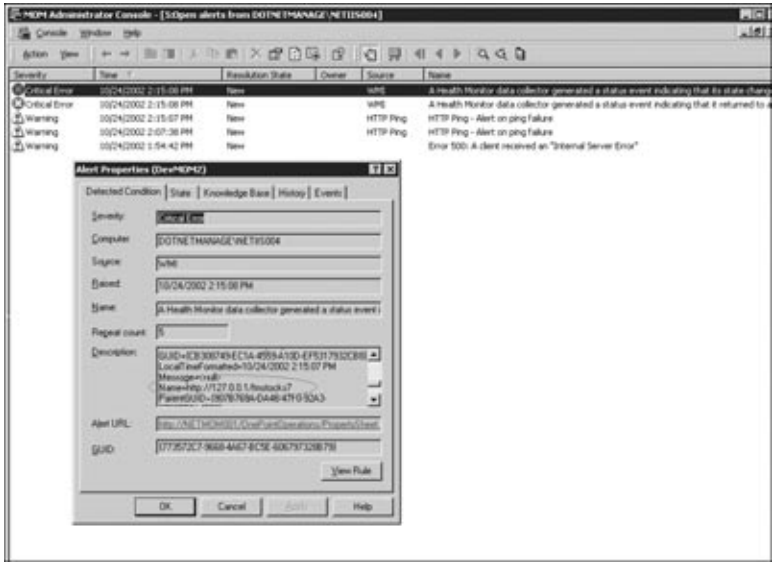


Figure 6.2
Errors reported by the Web server

Open one of the WMI critical errors and scroll though the description field. Note that the HTTP Monitor, which Health Monitor runs on the local computer, reported the error. Launch Application Center on the local computer (NETIIS004), and you can see that the HTTP Monitor generated a critical error.

► **To configure Microsoft Operations Manager to create an e-mail notification**

1. On the computer on which you installed the MOM Administrator console, click **Start**, point to **Program Files**, point to **Microsoft Operations Manager**, and then click **MOM Administrator Console**.
2. In the left-hand pane of the **MOM Administrator Console**, expand the **Configuration** node, select **Global Settings** in the right-hand pane, and then double-click **Email Server**.

   The **Configuration Group Global Settings** dialog box appears.
3. Under **Transport**, make sure that you have either the **Microsoft Exchange** or **SMTP** messaging type selected, and then do one of the following:
   - If you have **Microsoft Exchange** selected, enter the name of the computer running Microsoft Exchange under **Exchange server**. Then type the name of the **Mailbox** on that computer that the MOM Service will use to send e-mail to Exchange recipients.

     Note: To use Exchange as the transport method, you must log on to the computer that runs the MOM Service as the MOM Service Account, and then create a MAPI profile for the mailbox name you entered.
   - If you have **SMTP** selected, enter the name of the SMTP computer in the **Server name** field, the **Return address** (in the form *user@domain*) for notifications, and the **SMTP Port** number (the default is 25).
4. Click **OK** to close the **Configuration Group Global Settings** dialog box.
5. Expand the **Rules** node in the left-hand pane, and then open the **Processing Rule Group** that you set up for monitoring synthetic transactions. In the example in Chapter 5 under the "Monitoring Synthetic Transaction Events," this was called **Web Monitor Synthetic Transactions – PRG**.
6. Expand **Web Monitor Synthetic Transactions – PRG,** and then right-click **Event Processing Rules**.

   In the right-hand pane, you should now see the **Web Monitor Failure** event processing rule that you created in Chapter 5.
7. Double-click **Web Monitor Failure**.

   The Event Processing Rule Properties dialog box appears.
8. On the **Responses** tab, click the **Add** button, and then select **Send a notification to a notification group**.

   The **Send a Notification to a Notification Group** dialog box appears.

9. On the **Notification** tab, select the **Notification group** from the drop-down list that you want informed when this event occurs.

   If no suitable notification group appears, you can create a new notification group or new operator by clicking the **New** button.

10. On the **Email format** tab, either accept the default e-mail format or click **Custom email format** and amend the e-mail message as necessary. You can use the buttons to the right of the **Subject** and **Message** fields to add message variables as appropriate.

11. If you are also configuring paging for this event, click the **Page Format** tab and either accept the **Default page format** or create a **Custom page format** message.

12. If you want this event to generate a network message, click the **Command Format** tab and either use the **default command** or create a **Custom Command** to carry out a particular action.

    Note: To configure the default command format, amend the relevant value on the **Notification Command Format** tab under **Global Settings**.

13. Once you have finished configuring notifications, click **OK** twice to close the dialog boxes.

Defining and Generating Reports

Compared to the excitement and immediacy of alerts, reports might not seem interesting. When you think of reports, does this summon up images of dashing superheroes flying in to save your service from imminent collapse? Or does it remind you of the irksome kid at school who always produced perfect coursework and nobody liked? Putting it bluntly, reports have a bit of an image problem. They're just not that exciting.

But how would you rather spend your free time? Relaxing by your favorite river, fishing rod in hand? (Insert alternative ideal relaxation scenario here.) Or would you rather be trying to restart unresponsive applications in a remote data center, with your boss screaming in your ear about lost revenues while your Sunday lunch gets cold?

Although alerting frameworks are part of any monitoring solution that demands high levels of availability, alerting is by its very nature a reaction to something that has either failed or is about to fail, usually at the most inconvenient time. But reporting, if implemented correctly, allows you to take the initiative with issues that might appear in the future, which allows you to make corrections before the problem affects your SLA and, more importantly, your free time.

Finally, don't forget two other vital functions of reporting—demonstrating what a good job you are doing at the moment and justifying further expenditure. Without detailed logs showing the handling of automatic recoveries, as well as the time it takes to restore services, how are you going to justify expenditure on that fantastic new support tool?

General Principles

The reporting information you provide must be:

- Accurate
- Timely
- Relevant
- In a suitable format
- Automatically generated

Accuracy

Reports need to be representative rather than scientifically accurate. Identifying trends is more important that gauging usage to one thousandth of a percent. For example, you probably are not too worried about whether your processors are averaging 88.6 percent or 88.7 percent utilization. You are more likely to be concerned about why your processors are unexpectedly busy. Accuracy to two significant figures (giving an error of 1 percent) is normally acceptable.

Timeliness

Reports need to be timely, although not to the same degree as alerts. Reports must be recent enough to support decision making and to identify trends before they affect application operations.

Relevance

Reporting information must be relevant to the people reading the reports. Management reports do not show detailed application operations; however, your technical reports definitely will.

Suitable Format

Report format requirements will vary from organization to organization. MOM lets you change report formats using Microsoft Access 2000.

Automatic Report Generation

Implement a system that creates reports automatically, rather than manually. If your management applications publish reports onto the corporate intranet, this meets your requirements for timeliness, relevance, and format suitability. Automatic

reporting to a Web site reduces costs, increases reporting efficiency, saves trees, and makes your life a lot easier. Automatic report publishing is a central feature of the reporting module in MOM.

Creating the Reporting Environment

When you implement an effective reporting system, you must understand why and for whom you are producing your reports. For example, if your CEO wants to know how the performance of the company's stock-trading service, he or she is probably not looking for a printout of processor usage against time for each computer on the network.

Reporting information tends to fall into at least three categories:

- Reporting for decision makers
- Reporting on service levels
- Reporting for technical analysis

Reporting for Decision Makers

When reporting for decision makers, your main task is to provide the information your executives want in the format they want it. Reporting at this level looks at the high-level picture, concentrating on areas such as total uptime, service loading, transactions per second, data errors, and measurements that directly relate to profit or to organizational functioning.

You can link reporting categories at this level to MOF or the UK-based Office of Government Commerce IT Infrastructure Library (ITIL) definitions. Examples of executive level reports include:

- Availability management
- Financial management
- Capacity management
- Incident management
- Service continuity management
- Problem management

For a full listing of the service management functions, see the MOF Resource Library, at *http://www.microsoft.com/technet/itsolutions/tandp/opex/mofrl/default.asp.*

Reporting for Service Level Agreements

Reporting for SLAs (or service level management) differs from other reporting types because SLA reports tend to be delivered for an external audience, that is, the service customer. In a hosting environment, customers are the external organizations paying for the hosted service. With internal SLAs, the audience consists of the other departments within the organization.

Reporting for SLAs covers the following management areas:

- Availability management
- Service continuity management
- Capacity management

Reporting for Technical Analysis

Reports for technical consumption deal with information and trends that do not appear in the management reports. These reports focus on spotting the elusive memory leak, predicting when a volume will fill up, keeping tabs on your network traffic, or calculating how many instances of an application can run on a computer. This helps you specify when to request more storage, at what point to migrate to a Gigabit Ethernet, or when to insist that your developers sit down and fix application issues.

Reporting for technical analysis covers the widest range of MOF service management functions, including:

- Incident management
- Problem management
- Availability management
- Service continuity management
- Capacity management
- Configuration management
- Security administration
- System administration
- Network administration
- Directory services administration
- Storage management
- Print/Output management

Reporting on Specific .NET Framework Issues

You probably already have reports providing technical analysis of your existing Windows-based applications. Most of these reports work very well with .NET-based applications. However, two areas specific to .NET-based applications are:

- Interpreting application memory consumption
- Reporting on instrumented applications

Interpreting Application Memory Consumption

Memory consumption in .NET-based applications differs from Windows-based applications because the .NET Framework provides dynamic garbage collection. Hence interpreting memory consumption reports is slightly different.

Windows DNA applications that use ASP run either in the InetInfo.exe process or in Dllhost.exe, depending on the IIS virtual directory isolation level. With ASP.NET applications, the Aspnet.exe process does the work, and there can be more than one instance of Aspnet.exe.

Memory allocation behavior has changed compared to other Windows-based applications. .NET-based applications take advantage of the .NET Framework dynamic garbage collector, which produces differences in the behavior of the **Memory: Private Bytes** metric. Windows-based applications created using a language like Visual C++ must explicitly release memory and objects. With .NET-based applications, expect to see linear growth in **Private Bytes** until the dynamic garbage collector runs.

When the garbage collector runs, **Private Bytes** drops precipitously. When reporting on a processes' memory allocation, report over a long time to smooth out the bumpy nature of memory consumption in .NET-based applications.

Reporting on Instrumented Applications

Enterprise Instrumentation Framework (EIF) can extract additional monitoring information from your .NET-based application, as described in Chapter 4, "Instrumenting .NET-Based Applications." Instrumentation can also help you troubleshoot application problems.

For example, Fitch and Mather added custom instrumentation to FMStocks to trace Sellstock transactions through the Presentation, Business, and Data tiers. Tracing does not take place during normal operations, but when problems arise, the operations team edits the **TraceSessions.config** file to enable tracing. Your development team must provide you with custom reports to process this information for this part of your technical reporting.

For more information on the instrumentation example, see the Request Trace Sample in Appendix A of Chapter 4.

Reporting on .NET-Connected Applications

When managing .NET-connected applications, the line between operations management (as described by the MOF) and service-level management starts to blur. .NET-connected applications are a special case of .NET-based applications that use Web services, so you could report on application operation as you would with other .NET-based applications. However, .NET-connected applications provide high-level services to your enterprise, so the important reports for .NET-connected applications involve service levels.

So how does the consumption of Web services come into play? To answer that question, examine the structure of a .NET-connected application through the following example.

Fitch and Mather plans to turn the FMStocks application into a .NET-connected application. Fitch and Mather determines that customers want to have up to the minute stock prices when they run an account balance inquiry. To get these stock prices, Fitch and Mather engages an outside company to provide stock price information as a Web service. Figure 6.3 shows a block diagram of the FMStocks account balance transaction.

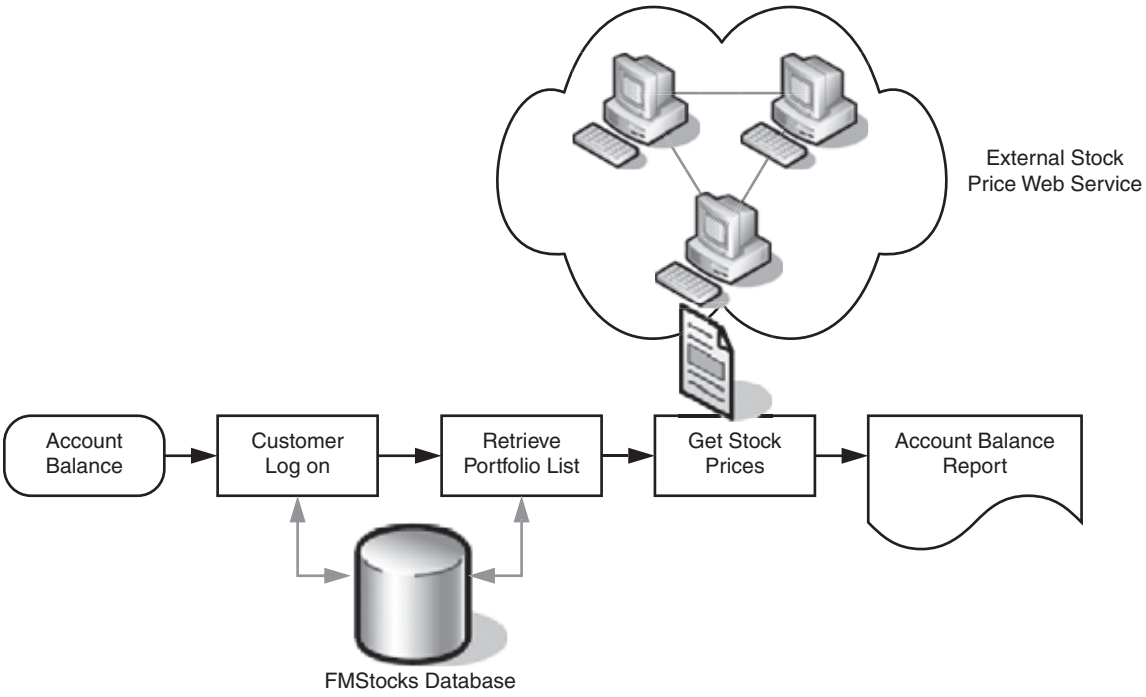


Figure 6.3
FMStocks account balance with external Web service

Fitch and Mather then implements a separate corporate customer relationship management (CRM) system and, rather than have the FMStocks application maintain its customer list, Fitch and Mather wants FMStocks to use the corporate CRM system for user account validation. To accomplish this, Fitch and Mather deploys an internal Web service for account validation. Figure 6.4 shows the modified block diagram for the FMStocks account balance transaction.

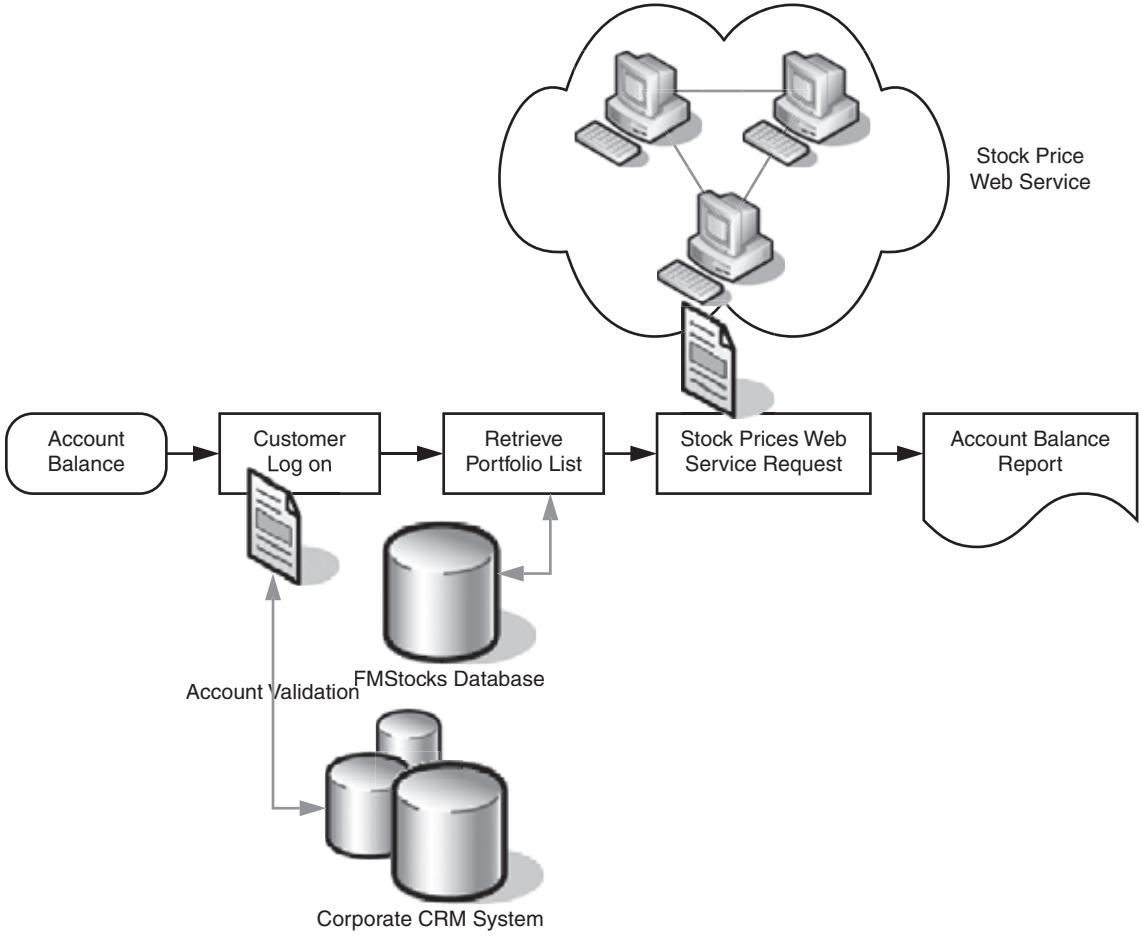


Figure 6.4
FMStocks account transaction with internal Web service

By now, you may be thinking that Web services consumption has not changed the application much, and are wondering why a .NET-connected application is any different from a .NET-based one. From the logical perspective, little separates them, but operationally, there are significant differences.

First, consumption of Web services by the FMStocks application helps you identify key indicators of overall application health. If these Web services are unavailable, the FMStocks application will also be unavailable, or in other words, unhealthy. Second, because these are Web services, you can monitor them more easily than if they were deeply embedded within the internal application logic.

Reporting on Synthetic Transactions

To create reports on service levels, use synthetic transactions executed against the Web service. To do this, employ monitoring applications, such as Web Monitor or Cluster Sentinel. Create reports that cover areas such as average response time and total uptime for the Web service. When you track and report these statistics over time, you record overall availability of this Web service.

You can implement synthetic transactions for reporting just like synthetic transactions for monitoring. Use the same synthetic transaction tools as in the other areas of your health monitoring solution. Figure 6.5 shows synthetic transactions reporting on Web service availability.

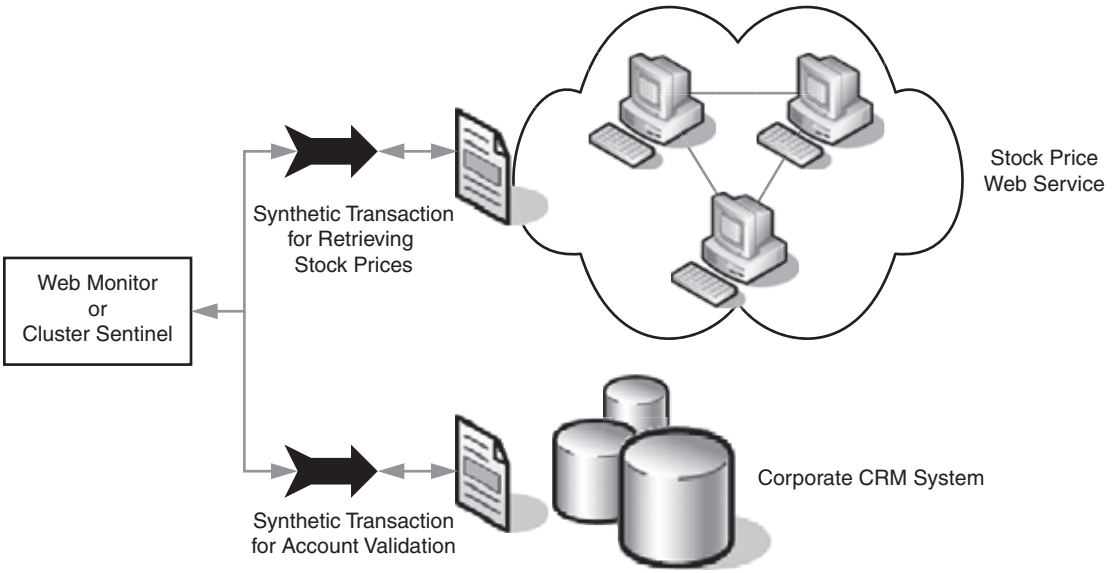


Figure 6.5
Using synthetic transactions to report on Web service availability

Guaranteeing operational health involves reporting on the availability of individual Web services that a .NET-connected application uses. This operational health picture overlaps with service-level management, because you must provide health and service-level reporting for the application services provided to your end users.

When reporting on application service levels, you must have a set of transaction scenarios that represent characteristic end-user interactions. The FMStocks account balance example is one such scenario. Build these scenarios into a set of synthetic transactions that execute against the application.

Building synthetic transactions usually requires third-party tools that are more sophisticated than Web Monitor or Cluster Sentinel. Use of these tools is outside the scope of this book.

Figure 6.6 shows how you can use synthetic transactions in FMStocks to check a dummy account balance.

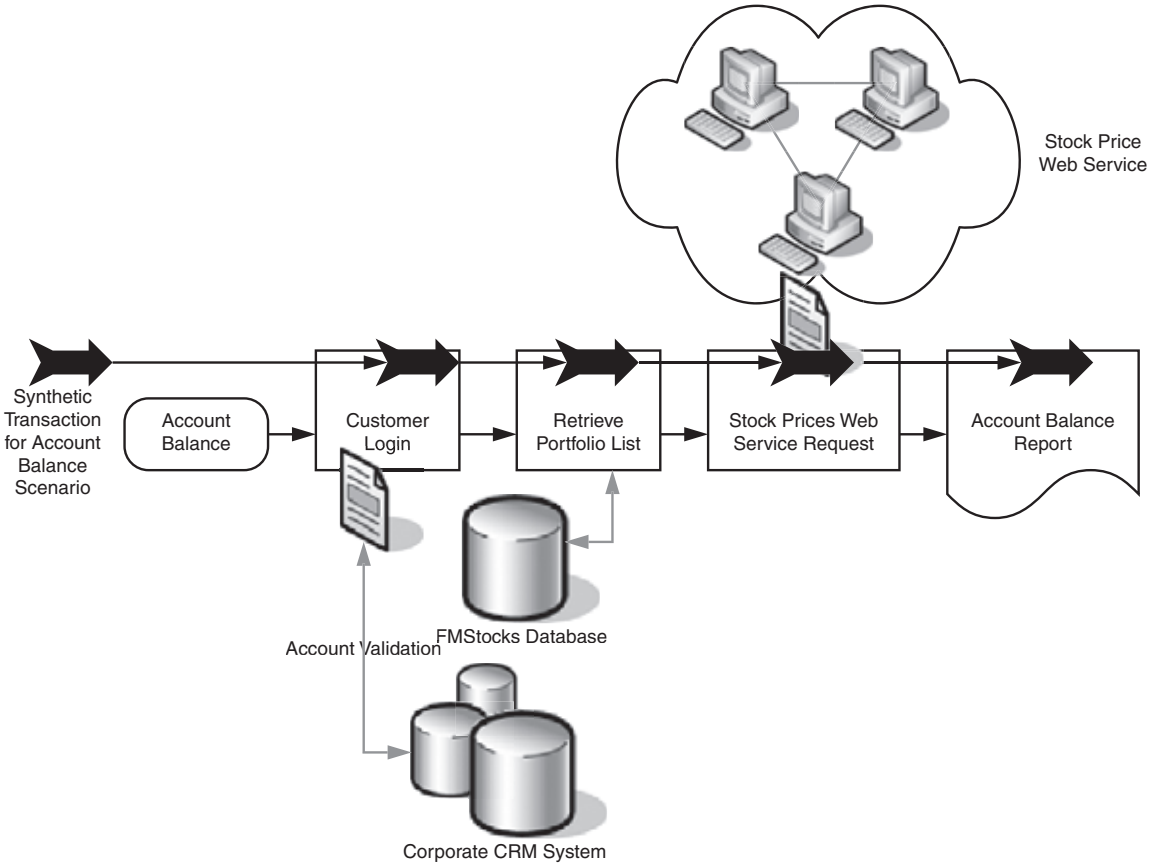


Figure 6.6
Synthetic Transactions for FMStocks Account Balance Scenario

Creating reports on the response time and availability of an entire transaction sequence helps provide service-level management for your .NET-connected application. Combine this with operational reporting on the individual Web services that the application consumes to create the basis for sound operational management of your .NET-connected application.

Reporting Resources

Ultimately, reports are not an end in themselves—you need to interpret their output and relate it to what is going on in your environment. Because of the relative newness of the .NET Framework, you should monitor the following technical references and newsgroups for news regarding operational issues:

- .NET Framework Newsgroup on the MSDN Web site at *http://msdn.microsoft.com /newsgroups/?url=/newsgroups/ loadframes.asp?icp=msdn&slcid=us&newsgroup=microsoft.public.dotnet.framework&frame=true*
- .NET Development Resources on the MSDN Web site at *http://msdn.microsoft.com /library/default.asp?url=/nhp/Default.asp?contentid=28000519*

You have now looked at reporting in general terms and in relation to .NET-based and .NET-connected applications. You can now look at specific reporting tools.

Reporting with Microsoft Operations Manager

You generate reports in MOM using a separate Access front-end. This allows you to create and view preconfigured reports or to generate custom queries and report formats. To run MOM Reporting, install Microsoft Access runtime on the same computer on which you installed the MOM Reporting application. You can install the Access runtime when setting up MOM.

Note: To edit reports, use a full version of Access 2000. Do not use Access 2002 to edit the MOM Reporting database because this prevents Access 2000 runtime from opening the database.

MOM has built-in automatic report-generating and publishing features, which makes it easy for you to create system-level reports and publish them on your intranet.

Viewing and Editing Reports

To view or edit reports in MOM, following these procedures.

► **To view reports**

1. Log on to the computer on which you installed the MOM Reporting component, click **Start**, point to **Programs**, point to **Microsoft Operations Manager**, and then click **MOM Reporting**.

   The Microsoft Operations Manager Reporting window appears.
2. In the left-hand pane, expand the list of **Available reports**, and then expand the **+** signs to see the full range of reports.
3. To view a specific report, click the report name in the left-hand pane.

   The Criteria for Selected Report window appears in the right-hand pane.

4. Under **Date Range**, select the dates for which you want the report to provide information.
5. Under **Chart Options**, choose from a **bar**, **line**, or **bar (3-D)** display type.
6. Optionally, select the **Y-axis range** check box, and then enter upper and lower values for the Y (vertical) axis. For example, if you are just interested in the range between 20 and 40 percent, enter **20 to 40**.

   A report appears that displays the requested statistics over the last week. To print this report, click the **Print** icon.

► To alter the current report layout

1. Click the **View** button at the top left-hand side of the toolbar, and then select **Design View** from the drop-down list. (Note that **View** is a tooltip and appears if you place your mouse over the correct button).
2. Change the format of the report and add or remove fields as necessary.
3. Click **Close** to return to the Microsoft Operations Manager Reporting window.

► To create multiple reports at the same time

1. In the **Microsoft Operations Manager Reporting** window, click **Print or View Multiple Reports** on the toolbar. The **Print Report Set** window appears.
2. In the **Print Report Set** window, select the reports that you want to create, and then click **Print**, **View**, or **Save as HTML**.

   If you click **Save as HTML**, this saves the report to the \Events\Reports\Reportname directory in the Microsoft Operations Manager default installation folder. The default Microsoft Operations Manager installation folder is C:\Program Files\Microsoft Operations Manager 2000.

Note: Installing MOM creates and shares the report directory, which you can find by connecting to \\MOMServer\Reports. However, the default permission on this share is **Everyone – Full Control**, which you probably want to change. Installing MOM also creates a folder on the default Web site at http://moms*servername*/reporting. This makes it easy to publish reports on your intranet.

► To change the default HTML reports folder

1. In the Microsoft Operations Manager Reporting window, click **HTML Properties** on the toolbar.

   The HTML Properties dialog box appears.
2. Under **HTML output root directory (required)**, type a new default report directory or click **Browse** to select the report directory target.
3. Click **OK** to accept the new report directory.

► **To make changes to the Access modules, tables, views, and queries**

1. On the toolbar, click **Enter Access Design Mode**.

   Note: It is recommended that you back up your current database before you make changes to reports using Access.

2. Make the necessary changes.
3. Close the Microsoft Operations Manager Reporting window.

Automating Reports

The MOM Reporting Module automates report generation and can either print or publish the reports as HTML. To configure automatic reporting in Microsoft Operations Manager, follow this procedure.

Note: The Microsoft Operations Manager (MOM) Service Pack 1 (SP1) introduces significant enhancements to the reporting capabilities of Microsoft Operations Manager.

► **To configure automatic reporting in Microsoft Operations Manager**

1. Log on to the computer on which you installed the MOM Reporting component, click **Start**, point to **Programs**, point to **Microsoft Operations Manager** and then click on **MOM Reporting**.

   The Microsoft Operations Manager Reporting window appears.
2. If necessary, change the default HTML directory as detailed in the previous procedure.
3. On the toolbar, click **Command Line Wizard**, or on the **Tools** menu, click **Report Command Line Wizard**.
4. When the **Reporting Command Line Wizard** welcome page appears, click **Next**.
5. On the **Select Schedule Period** page, select either **Last x Days** (where x can be a value between 1 and 999 days), **Weekly**, or **Monthly** as the period that you want covered in the report.

   Note that this does not schedule the report itself to run daily, weekly, or monthly.

   Optionally, if you want the reports to include only information within a specific range of times, such as between 08:30 and 09:30 then enter the times under **Desired Hour Range**. Otherwise, the report times include all times of the day.
6. Click **Next**.
7. On the **Specify Forest, Domains, Server and Chart Type** page, select from the following check boxes: **Forests**, **Domains**, **Servers** and **ChartTypes**. Type the relevant Forest, Domain, or Server name in the adjoining text box, and then click **Next**.

8. On the **Select Reports** page, click the report or reports that you want to run. You can select multiple reports by holding down the CTRL key and clicking the relevant report names.
9. To include a description of the report in the HTML file header, select the **Include report description in report header** check box, and then click **Next**.

   This description displays in the browser window when you view the report.
10. On the **Select Report Output Type** page, select **Printer** or **HTML** as the preferred destination, and then click **Next**.

    If you select HTML as the preferred output type, you then have the opportunity to archive the previous report. This report is saved in a subfolder with a date and time stamp for the folder name. This is useful for keeping old copies of reports.
11. In the **Select Data Source** dialog box, ensure that **Current data source** is selected, and then click **Next**.
12. When the Completing the Report Command Line Wizard appears, under **Batch file name**, check that the path and batch file name is correct, and then select the **Launch Scheduled Tasks Folder** check box.

The Report Command Line Wizard saves a batch file of commands to the specified location. Use the Scheduled Task Wizard to schedule this task for a specific time.

► **To schedule the task for a specific time**

1. In the Scheduled Task window, click **Add Scheduled Task**.
2. On the **Scheduled Task Wizard** page, click **Next**.
3. On the second page of the **Scheduled Task Wizard**, click **Browse**. Locate the batch file that you saved in the previous procedure.
4. Enter a name for the task. Use a descriptive name for the job, such as "Daily MOM Reports."
5. Select when you want the task performed.

   Normally, this will tie into the reporting period that you selected in the previous section. Hence if you are creating Microsoft Operations Manager reports for a 24-hour period, then the report generation task will be **Daily**.
6. Click **Next**.

Note: You can change the task timings once you have finished configuring the Scheduled Task Wizard.

---

7. Enter a **Start time** and **Start Date** for the task, and then click **Next** to continue.
8. Enter a user name and password combination with the permissions to run the task. This user account needs to be a member of the local group OnePointOp Reporting on the MOM computer. By default, Domain Admins and the Administrator account are members of this group.

9. Click **Next**.
10. To make further changes to the scheduled task, click **Open advanced properties for this task**.

    The **<Taskname>** dialog box appears.
11. Click the **Schedule** tab to change to the time that the reporting task will run, and click the **Settings** tab to change power and hibernation options.
12. Click **OK** to close the **<Taskname>** dialog box.

Microsoft Operations Manager will now create, publish, and archive reports automatically. To view a report, open http://*computername*/Reporting/Default.htm.

Reporting with Application Center 2000

Microsoft Application Center 2000 provides limited reporting on computers that are members of an Application Center cluster. However, these reports are part of the user interface, rather than database-ready or printable reports like those that MOM provides.

The coarse-grained health monitoring picture in Chapter 3, "Selecting Data Sources" illustrates a monitoring view in terms of raw system metrics, such as CPU and Memory, as well as information about IIS. Application Center provides a preconfigured view that shows this same information. Figure 6.7 shows this cluster view.

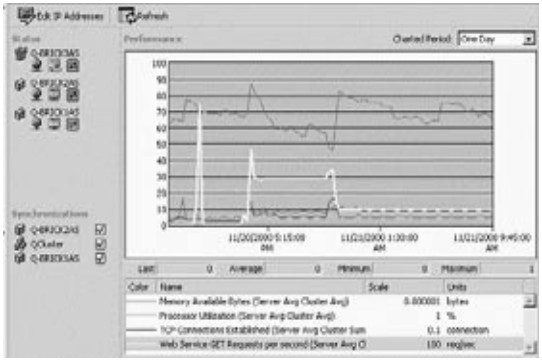


Figure 6.7
Application center cluster view

Reporting with AppMetrics

The Fitch and Mather scenario uses the Production Monitor from AppMetrics. The Production Monitor generates metrics on an interval basis about the activity of applications that use Enterprise Services. More specifically, Production Monitor calculates:

- Total values

- Rates of component activity
- Average durations

In Chapter 5, "Configuring Management Applications," you set up an AppMetrics Production Monitor to monitor the FMStocks application. You can now extract useful information from AppMetrics.

Note: AppMetrics reports require Microsoft Excel version 7.0 or higher.

AppMetrics has four measurement categories, but you are concerned only with the component reports category. Figures 6.8, 6.9, and 6.10 show the results of monitoring the FMStocks application with a reporting interval of four minutes and uploading the logged data to the database once per hour.

In this procedure, you use AppMetrics to generate reports from a production monitor.

► **To view the AppMetrics reports**

1. On the AppMetrics Console and Reports computer, click **Start**, point to **Program Files**, point to **Xtremesoft**, and then click **AppMetrics Reports**.

   This launches Microsoft Excel.
2. Select the **Always enable macros from this source** check box to enable the **Enable Macros** button, and then click **Enable Macros** in the initial Microsoft Excel dialog box.

   The AppMetrics Report dialog box appears.
3. On the **Manager** tab, enter the name of the AppMetrics Manager machine.
4. On the **Dataset** tab, enter the name of your Production Monitor.

   In the example from Chapter 5, the monitor name is **FMStocks Production**.
5. On the **Time Filter** tab, select the time frame for your report.
6. On the **Reports** tab, leave all of the reports selected, and then click **Go**.

   The **AppMetrics Components** dialog box appears. It displays a drop-down list of all components for which you have collected data.
7. To switch the displayed component class, select the class from the drop-down list, and then click **Refresh Charts**.

The component report consists of two sections. The top section charts the metrics describing the activity of a single component class. Figures 6.8, 6.9, and 6.10 on the next page show the activity charts for the FMStocks component called **FMStocks7.DAL.Broker**. Figure 6.8 shows the number of active components, Figure 6.9 shows the rate at which components start and complete, and Figure 6.10 shows the chart for the average transaction duration.

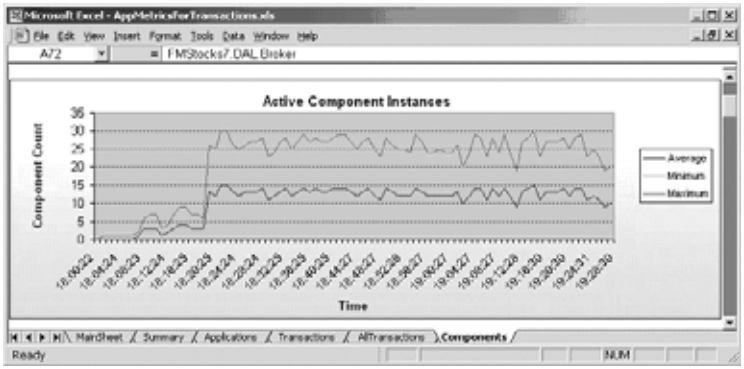


Figure 6.8
AppMetrics active components chart

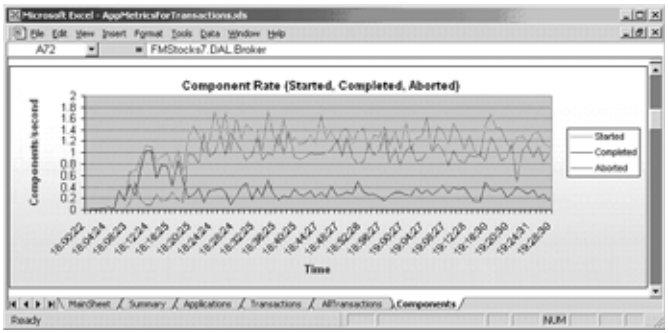


Figure 6.9
AppMetrics component rate chart

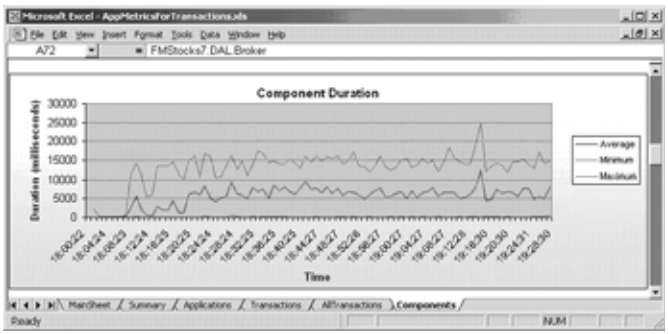


Figure 6.10
AppMetrics component duration chart

The Excel worksheet that follows the charts contains a table that shows all components monitored by AppMetrics and the measurement data for the selected period. Using this table, you can compare values using the built-in data analysis features of Excel.

In Chapter 5, "Configuring Management Applications," you also set up an AppMetrics Diagnostics Monitor to obtain detailed information about the relationships between Enterprise Services components. The relationships are described using the AppMetrics Drill-Down Report. The AppMetrics Drill-Down Report gets its information from an AppMetrics Diagnostics Monitor.

► **To view the AppMetrics Drill-Down Report**

1. Find and then open the file **AppMetricsForTransactionsDrillDown.xls**. This launches Microsoft Excel.

   Note: AppMetricsForTransactionsDrillDown.xls will be in the directory where the AppMetrics administrator put it.

2. In the initial Microsoft Excel dialog box, check the **Always trust macros from this source** check box, and then click the **Enable Macros** button. The AppMetrics Drill-Down Report dialog box appears.
3. On the **Server** tab, enter the name of the SQL Server computer, and then select **NT Authentication** or **SQL Authentication**.
4. The SQL Server computer is the same as the AppMetrics Manager computer, and you can use NT Authentication if you logged on using an administrative account.
5. On the **Database** tab, select the name of your Diagnostics Monitor database.
6. In the example from Chapter 5, the monitor name is **FMStocks_Diagnostics**.
7. On the **Time Filter** tab, select the timeframe for your report.
8. On the **Reports** tab, leave all of the reports selected, and then click **Go**.

To analyze the relationship between components, look at the **Methods** tab. The Transaction column shows you the first Enterprise Services component name that is called in the sequence of related method calls. The Method Name column lists the methods called in sequence. Using the relative start and end times, determine which components called which other components.

For more information, see the AppMetrics Drill-Down Reports documentation. The AppMetrics Drill-Down Reports PDF file is in the same directory as the Excel file in the previous set of steps.

Summary

In this chapter, you looked at how to configure notifications and reporting. You defined what appropriate actions are and linked those actions to alert severity and corresponding readiness levels. You discussed reporting and its importance, and you looked at some typical issues that can occur when reporting on components that use the.NET Framework. Finally, you saw how to configure your management applications to create reports.

Module 2

Securing .NET-Based Applications

7

General Security Recommendations

Introduction

In any medium to large scale enterprise, it is very likely that a significant number of applications will need to be supported. One of the most difficult challenges facing IT staff today is how to ensure that these applications run properly while maintaining the security of the environment.

To help ensure that .NET-based applications run securely in your environment, you need to consider two elements. First, you should examine the security of the code itself. For applications written internally you should follow documented procedures for writing secure code.

Note: For information on secure coding, see "Building Secure ASP.NET Applications", listed in the More Information section at the end of this chapter.

However, it is at least as important to ensure that the applications operate in a secure production environment. In this section, we focus explicitly on the operations required to create and maintain a secure environment on networks running .NET-based applications.

You should not use this section as a complete reference to cover all aspects of creating and maintaining a secure environment. Instead, our aim is to define security best practices that you can use as a basis for your overall security strategy for .NET-based applications. To lock down application servers we build on a technique pioneered in *Security Operations for Microsoft Windows 2000 Server* (Microsoft Press, ISBN: 0-7356-1823-2) and make further suggestions for servers running .NET-based applications, as well as showing how we specifically locked down the environment to support a particular application.

To allow you to read this section independently of other materials, we have reproduced some material from *Security Operations for Microsoft Windows 2000 Server*. However, if you have not read that book, you would be strongly advised to do so before implementing the suggestions made here.

In this chapter, we discuss Microsoft's general security strategy and then examine the importance of patch management for servers running .NET-based applications.

Get Secure and Stay Secure

In October 2001, Microsoft launched an initiative known as the Strategic Technology Protection Program (STPP). The aim of this program is to integrate Microsoft products, services, and support that focus on security. Microsoft sees the process of maintaining a secure environment as two related phases: Get Secure and Stay Secure.

Get Secure

The first phase is called Get Secure. To help your organization achieve an appropriate level of security, follow the Get Secure recommendations in the Microsoft Security Tool Kit, which can be accessed online (see the "More Information" section for details on the tool kit and the STPP).

Stay Secure

The second phase is known as Stay Secure. It is one thing to create an environment that is initially secure. However, once your environment is up and running, it's entirely another to keep the environment secure over time, take preventative action against threats, and respond to them effectively when they do occur.

Scope of this Guide

This guide is focused explicitly on the operations required to create and maintain a secure environment on servers running .NET-based applications. We examine two specific roles defined for servers — Presentation tier servers and Business tier servers.

You should use this guide as part of your overall security strategy, not as a complete reference to cover all aspects of creating and maintaining a secure environment. The diagram provides a high level view of these areas, the dark shaded box with white text is covered in this guide and the other shaded areas are covered in *Security Operations for Microsoft Windows 2000 Server*.

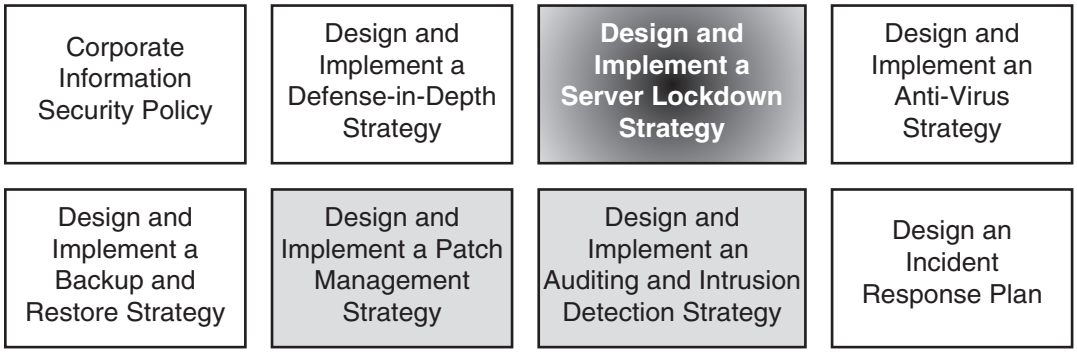


Figure 7.1
Scope of this guide in relation to your overall security strategy for .NET-based applications

Note: *Security Operations for Microsoft Windows 2000 Server* is available online. For further details, see the "More Information" section at the end of this chapter.

The diagram on the next page shows the steps required to help make a server secure (Get Secure) and help keep it that way (Stay Secure). It also shows how the chapters of this guide and *Security Operations for Microsoft Windows 2000 Server* will help you achieve those aims.

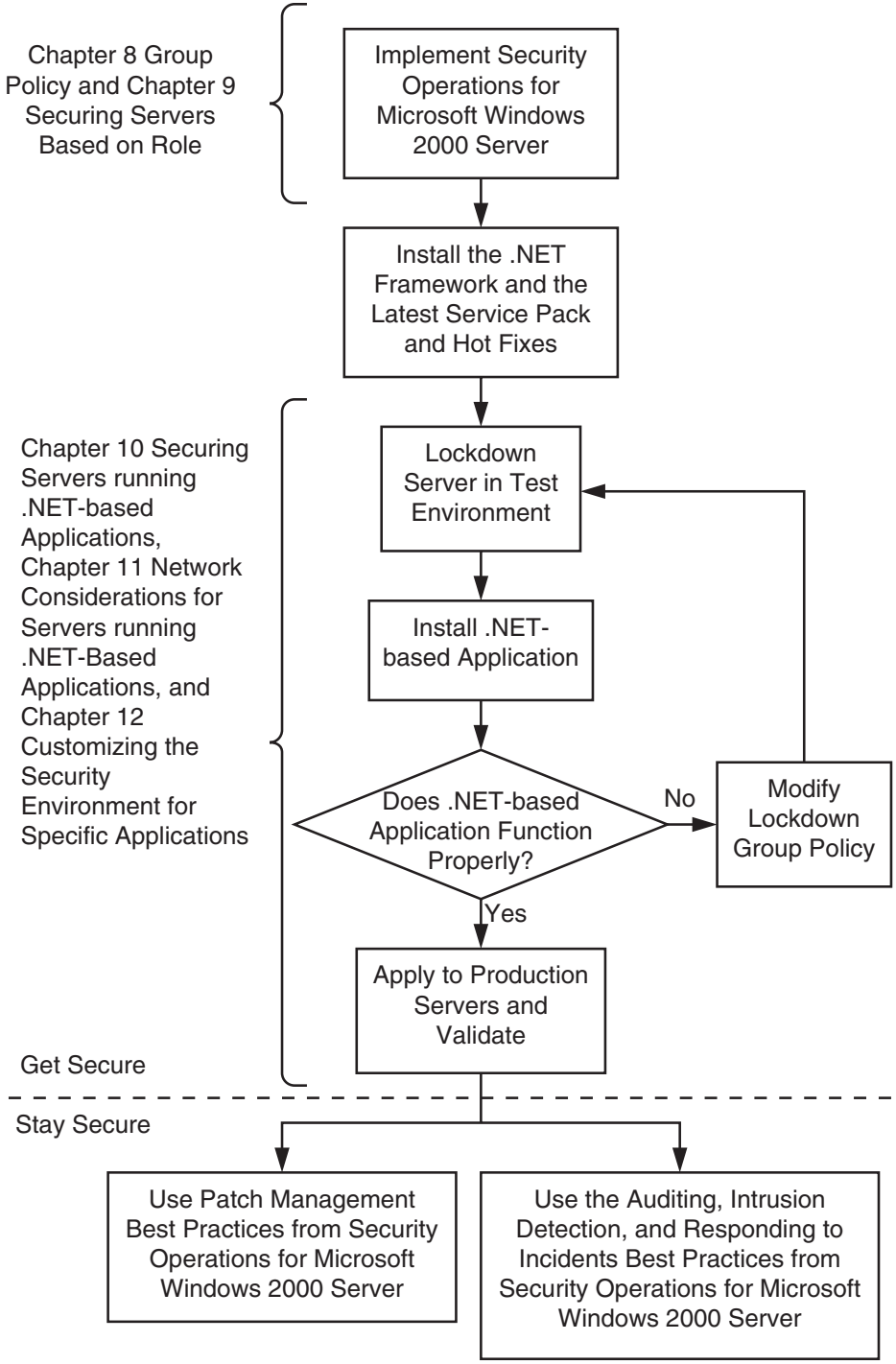


Figure 7.2
Process flowchart showing Get Secure and Stay Secure phases

Patch Management Strategies

When you deploy an application based on the .NET Framework, you must ensure that the clients and the servers accessing, running, and supporting the application are up to date with the latest hot fixes, security patches, and service packs. To ensure that both client and servers are current, you will need an effective patch management strategy, which covers assessment, testing, and deployment of new patches, when they become available. At the very least, as part of your patch management strategy, you should do the following:

- *Analyze the environment for missing service packs, patches, or hot fixes.* Look at the current environment and potential threats. Determine the patches you must deploy to reduce threats to your environment.
- *Plan the deployment of the service packs, patches, or hot fixes*. Determine which patches should be deployed to deter potential threats and vulnerabilities. Identify who will perform testing and deployment, as well as the steps involved.
- *Test service packs, patches, and hot fixes.* Review available patches, categorize them for your environment, and test them to make sure they work within your environment without any negative side effects. Understand what each patch does and how it affects your environment. Verify that each patch performs as planned.
- *Deploy the service packs, patches, or hot fixes.* Deploy the necessary patches to make your environment secure.
- *Monitor the updated systems.* Check all systems after deploying patches to make sure there are no undesired side effects.
- *Review for new service packs, patches, or hot fixes.* As part of an ongoing process, routinely review new service packs, patches, and hot fixes, as well as your environment to see which updates are needed. If, during the review, you find new patches are needed, return to the first step.

Note: For more details on patch management, see Chapter 5, "Patch Management," in *Security Operations for Windows 2000 Server*.

Determining Which Patches to Apply

To help you determine which service packs, security updates, or hot fixes must be applied to your client computers and servers, you can use the Microsoft Baseline Security Analyzer (MBSA).

MBSA uses an Extensible Markup Language (XML) database (downloaded from Microsoft) that contains a list of the latest hot fixes and service packs for the following:

- Windows NT 4.0
- Windows 2000
- Windows XP

- All system services, including IIS 4.0 and 5.0
- SQL Server 7.0 and 2000 (including Microsoft Data Engine)
- Internet Explorer 5.01 and later

MBSA allows you to check if the current configuration of your servers is up to date and has all the appropriate security patches. MBSA can use either the downloaded XML file or use a local XML database if you do not have a connection to the Internet.

Note: Wherever possible, you should avoid using a local XML database with MBSA as you risk relying on out of date results.

MBSA examines several values before it reports on the status of a patch. The registry key that is associated with the patch is checked first. If this registry key does not exist on your computer, the patch is considered not installed. If the key does exist, MBSA then examines the file version and file checksum for each file in the hot fix. If all of these values match, the patch is considered installed, but if any one of the tests do not work, the patch is either considered not installed, or it is flagged as a warning (depending upon the results of the check).

Note: To use MBSA, you must have administrator rights on the computer being checked for patches.

If you are using MBSA to verify your patch status, you should ensure it runs regularly. In most environments, the best way to do this is to schedule it.

In addition to testing for hot fixes and service packs, MBSA provides additional capabilities, including the following:

- Examining Windows-based desktops and servers for common security best practices, such as strong passwords.
- Scanning servers running IIS and SQL Server for common security configuration problems.
- Checking for poorly configured security zone settings in Microsoft Office, Outlook and Internet Explorer.

Two versions of MBSA exist: a graphical version and a command-line version, mbsacli.exe. The command-line version of MBSA can be used to script the functionality provided by the graphical version of MBSA.

Note: See the More Information section at the end of this chapter for further details on MBSA.

Deploying Service Packs and Hot fixes

Once you determine what service packs and hot fixes are required, you should thoroughly test them, ensuring that they install correctly in your test environment without causing any problems. Assuming that testing proceeds smoothly, you will now be in a position to install then in your production environment. Separate plans should be developed for the deployment of service packs and hot fixes.

Deploying Service Packs

Depending on the scenario you choose, you can use one or more of the following deployment tools and files:

- *Microsoft Systems Management Server (SMS)*. SMS provides a variety of tools to help you deploy the latest Windows 2000 service pack. With SMS v2.0 software distribution, you can automatically upgrade all SMS client computers in your organization with a new service pack. You can allow your users to run the service pack installation whenever they like, or you can schedule installation to run at a specific time. You can also schedule it to run on SMS client computers when users are not logged on.

Note: SMS provides tools for upgrading your current computers, but not for the installation of new computers that do not have an operating system already installed.

- *Microsoft Windows Installer and Group Policy*. Group Policy may be used by deploying the Update.msi file to computer accounts in the organization unit (OU) where the Update.msi file is assigned.

Deploying the Service Pack with SMS

You can use SMS to install the latest service pack from a shared distribution folder on the network. This method installs the service pack on SMS client computers already running the Microsoft Windows® 2000 operating system.

Note: For detailed information on installing Windows 2000 Service Pack 3 with SMS, see the Windows 2000 Service Pack 3 Installation and Deployment Guide, listed in the More Information section at the end of this chapter.

Deploying the Service Pack with Group Policy

Each Windows 2000 service pack includes a Windows Installer package file (Update.msi). The Update.msi file includes all the information required by Windows Installer.

By assigning the Update.msi file to computer accounts in the appropriate OU, you can ensure that all Windows 2000 computer accounts in that OU have the latest service pack. Windows Installer installs the service pack automatically when a user

starts his or her computer. The user is not given the choice whether or not to install the service pack.

Note: Microsoft does not support the use of user-based Group Policy deployments with Update.msi. User-based Group Policy deployments apply on a per-user basis. The installation of a service pack applies to the entire system.

Deploying Hot Fixes

Hot fixes are interim updates that address critical problems for which no feasible workaround is available or security issues. The hot fixes can be deployed using the following methods:

- Manually
- Microsoft Software Update Services (SUS)
- SMS

When you run the hot fix package, it automatically installs the updated system files and makes the necessary registry changes. Typically, the computer must be restarted after installation to ensure that Windows 2000 runs with an updated file set.

Manual Installation of Hot Fixes

You can manually download the hot fixes recommended by MBSA from the Microsoft TechNet HotFix & Security Bulletin Service (see the More Information section for details). Once the hot fixes are downloaded, manually install them by running the Hotfix.exe program. This program extracts the hot fix files, then executes the Update.exe program.

If multiple hot fixes are installed in succession, you can prevent multiple reboots by chaining the hot fixes. If you install multiple hot fixes at once and they perform multiple updates to a single file, hotfix.exe will automatically determine the correct version of the file to keep.

Note: If you are applying hot fixes to Windows NT 4.0 or pre-Windows 2000 Service Pack 2 hot fixes, you must use the QChain.exe tool to ensure that the correct file version is maintained. For more information on the QChain.exe tool, see Q314902: "HOW TO: Use QChain.exe to Install Multiple Hotfixes in Windows".

Microsoft Software Update Services

Software Update Services (SUS) extends the Windows Update technology, allowing an organization to manage the selection and distribution of hot fixes to the following operating systems:

- Microsoft Windows 2000 (All versions)
- Microsoft Windows XP (All versions)
- Microsoft Windows .NET Server (All versions)

SUS is composed of both client- and server-side applications. The client application is based on an updated version of the Windows Automatic Updates technology.

The server side SUS application is based on the same back-end technology as the public Windows Update site. It runs on a Windows 2000-based server with Service Pack 2 or later, that has IIS enabled.

Note: For more information on deploying Microsoft SUS, read the Software Update Services Deployment white paper, listed in the More Information section at the end of this chapter.

Deploying Hot Fixes with SMS

With the release of the SMS 2.0 Feature Pack, you can use SMS to deploy the latest hot fixes to client computers and servers. SMS uses the following process:

1. The Software Update Inventory Installer, which you install on the SMS site server, uses SMS software distribution features to deploy the Software Update Inventory Tool and the Software Update Sync Tool to the appropriate client computers.
   - The *Sync Tool* automatically downloads the latest inventory tools and database or catalog of software updates (XML file) on a regular basis and distributes them to the computers in your enterprise using SMS Distribution Points.
   - The *Scan Tool* uses the inventory tools and software update database provided by the Sync Tool to collect software update inventory data from your client computers. It then converts that data into a format compatible with the SMS inventory database and forwards the information to the SMS inventory database during regularly scheduled hardware inventory cycles.
2. SMS and the Feature Pack tools use the converted inventory data to determine which hot fixes are installed and which are missing from your client computers.
3. The Distribute Software Updates Wizard, which you install on the SMS site server, uses the inventory information to compile groups of related software updates. It provides you with the opportunity to:
   - Review and authorize updates for installation.
   - Download authorized updates and installation information.
   - Specify how updates install on the client computers.
   - Create SMS packages, programs, and advertisements to distribute updates to client computers on a schedule using SMS Distribution Points.
   - Deploy the Software Update Installation Agent to clients through a program included with the update advertisement package.
4. The client computers use a program created by the Distribute Software Updates Wizard to install the Software Updates Installation Agent. The Installation Agent processes the advertisements sent by the Distribution Wizard and facilitates the installation of authorized software updates.

5. Inventory and software update status information is forwarded to the SMS Inventory database on a regular cycle following any changes.
6. The Web Reports Add-In for Software Updates uses inventory and status information to create Web reports you can use to track progress of update inventory, distribution, and installation in your enterprise. You can view these reports through the SMS Web Reports Viewer in your browser.

Note: For more details on using SMS to distribute hot fixes to client computers in your organization, see the More Information section for a link to the SMS "Software Update Services Feature Pack".

Comparing SUS and SMS for Hot Fix Deployment

Both SUS and SMS may be used to distribute critical updates for Windows 2000– and Windows XP-based computers within your organization. The following table highlights differences between SUS and SMS in the area of deploying critical updates:

Table 7.1: Differences between SUS and SMS with the Feature Pack

Installation and Distribution Feature	SUS	SMS with the Feature Pack
Content	Built-in synchronization with the Windows Update site	Automatic download of necessary updates
Targeting	Basic targeting in which a client computer will receive all applicable patches from the SUS server to which it is assigned	Granular targeting based on criteria such as inventory, groups, OUs, and subnets
Geographic Distribution	SUS can download hot fixes from the Internet or from other SUS servers on the network	Site-to-site distribution that can be scheduled and is sensitive to wide area network (WAN) links
Installation	Installation can be controlled using Group Policy, registry settings, or manually	Installation can be manual or implement advanced scheduling
Status	Status is reported in the IIS logs	Status is reported with built-in filters and reporting

For small to medium sized organizations, SUS will generally be a suitable tool for hot fix deployment. However, if you have a large scale organization, you should consider other solutions with more advanced option, such as the SMS Feature Pack.

Summary

This chapter has introduced the Strategic Technology Protection Program (STTP). It has also shown the steps you can take to ensure that your environment is kept fully up to date on hot fixes and patches. You should use these guidelines to help you build an effective patch management strategy for your organization

More Information

Building Secure ASP.NET Applications

http://msdn.microsoft.com/library/en-us/dnnetsec/html/secnetlpMSDN.asp?frame=true

For more detail on how MOF can assist in your enterprise:

http://www.microsoft.com/mof

Information about the Strategic Technology Protection Program:

http://www.microsoft.com/security/mstpp.asp

Microsoft Security Tool Kit:

http://www.microsoft.com/technet/security/tools/stkintro.asp

Information on the Microsoft Security Notification Service:

http://www.microsoft.com/technet/security/bulletin/notify.asp

HotFix & Security Bulletin Service

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/current.asp

Security Operations Guide for Windows 2000 Server

http://www.microsoft.com/technet/security/prodtech/windows/windows2000/staysecure/default.asp

Chapter 5, "Patch Management," in the Security Operations Guide for Windows 2000 Server.

http://www.microsoft.com/technet/security/prodtech/windows/windows2000/staysecure/default.asp

Using SMS 2.0 to Deploy Security Tool Kit Fixes

http://www.microsoft.com/smserver/techinfo/deployment/20/deployosapps/smsseckit.asp

Securing IT with Systems Management Server

http://www.microsoft.com/smserver/evaluation/overview/secure.asp

Software Update Services Deployment White Paper

http://www.microsoft.com/windows2000/windowsupdate/sus/susdeployment.asp

Software Update Services Overview White Paper
http://www.microsoft.com/windows2000/windowsupdate/sus/susoverview.asp
Windows 2000 Service Pack 3 Installation and Deployment Guide
http://www.microsoft.com/windows2000/downloads/servicepacks/sp3/spdeploy.htm
Windows 2000 Hotfix Installation and Deployment Guide
http://www.microsoft.com/windows2000/downloads/servicepacks/sp3/hfdeploy.htm
SMS "Software Update Services Feature Pack"
http://www.microsoft.com/smserver/evaluation/overview/featurepacks/suspack.asp
Microsoft Baseline Security Analyzer
http://www.microsoft.com/technet/ security/tools/tools/mbsawp.asp
Windows 2000 Service Pack 3 Installation and Deployment Guide
http://www.microsoft.com/windows2000/downloads/servicepacks/sp3/spdeploy.htm

8

Managing Security with Windows 2000 Group Policy

Introduction

After you have determined the level of risk appropriate for your environment and established your overall security policy, it is time to start securing your environment. In a Windows 2000-based environment, this is mainly achieved through Group Policy.

In this chapter we will show how to set up Group Policy objects (GPOs) with security templates to define security settings in your Windows 2000-based environment and we will discuss a simple OU structure that will support the use of these GPOs.

Warning: Before implementing the security templates discussed in this chapter in a production environment, you must first test the security templates thoroughly in a lab to ensure your servers continue to function as expected.

Importance of Using Group Policy

The goal of security policies is to define the procedures for configuring and managing security in your environment. Windows 2000 Group Policy can help you to implement technical recommendations in your security policy for all the workstations and servers in your Active Directory domains. You can use Group Policy in conjunction with your OU structure to define specific security settings for certain server roles.

If you use Group Policy to implement security settings, you can ensure that any changes made to a policy will apply to all servers using that policy and that new servers will automatically obtain the new settings.

How Group Policy is Applied

To use Group Policy safely and efficiently, it is very important to understand how it is applied. A user or computer object can be subject to multiple GPOs. These are applied sequentially, and the settings accumulate, except in the case of a conflict, where, by default, settings in later policies override those in earlier ones.

The first policy to be applied is the local GPO. Every computer running Windows 2000 has a local GPO stored on it. By default, only nodes under Security Settings are configured. Settings in other parts of the local GPO's namespace are neither enabled nor disabled. The local GPO is stored on each server in %systemroot%\System32\GroupPolicy.

After the local GPO, subsequent GPOs are applied at the site, domain, parent OU and finally child OU. The diagram shows how each policy is applied:

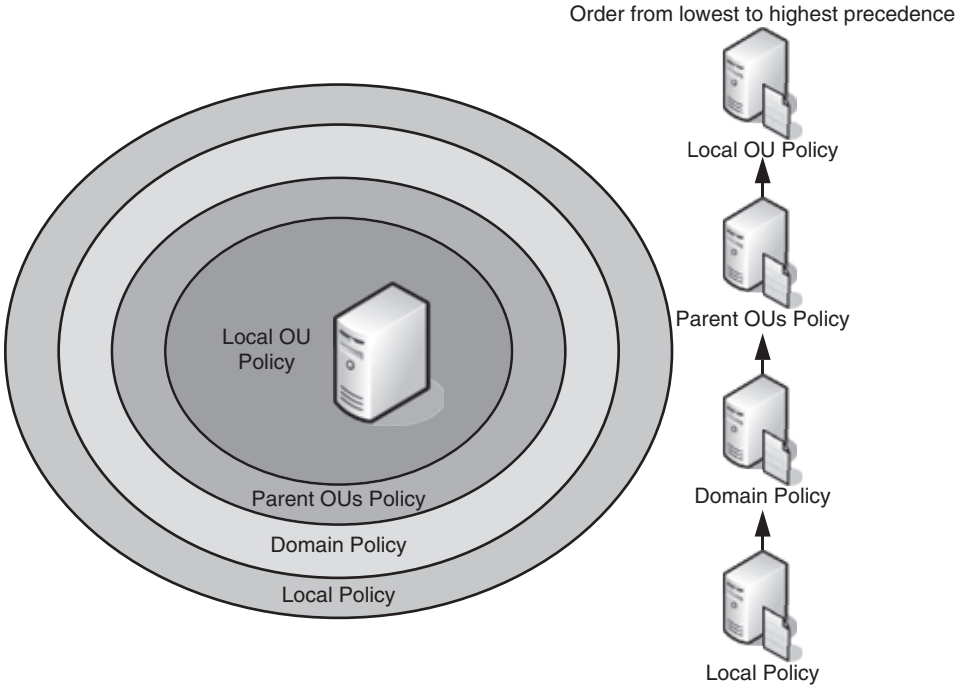


Figure 8.1
GPO application hierarchy

If there are multiple GPOs defined at each level, an administrator will set the order in which they are applied.

A user or computer will apply the settings defined in a Group Policy if a) the Group Policy is applied to their container and b) they appear in the Discretionary Access Control List (DACL) for the GPO with at least **Apply Group Policy** permission.

Note: By default, the built-in group, Authenticated Users, has the **Apply Group Policy** permission. This group contains all domain users and computers

Ensuring Group Policy is Applied

Group Policy settings are located (in part) in Active Directory. This means that changes to Group Policy are not applied immediately. Domain controllers first need to replicate Group Policy changes to other domain controllers. This will take up to 15 minutes within a site and significantly longer to replicate to other sites. Once changes have been replicated, there is a further time period (five minutes for domain controllers and 90 minutes plus or minus an offset of 30 minutes for other computers) before the changes in the policy are refreshed on the destination computer.

If you wish, you can force either of these actions to occur immediately.

► To force domain controller replication

1. Open Active Directory Sites and Services, expand Sites, expand the <site name>, and then expand Servers.
2. Expand both <DC name 1> and <DC name 2> and then, for each server select NTDS Settings.
3. In the right pane, right-click the connection object name and select Replicate Now. This will force replication immediately between both domain controllers.
4. Repeat steps 2 and 3 for each domain controller.

► To refresh policy manually on a server

At the server command prompt, type **Secedit /refreshpolicy machine_policy /enforce**. This command tells the server to check Active Directory for any updates to the policy and, if there are any, to download them immediately.

► To verify the effective policy settings

1. Start Local Security Policy.
2. Under Security Settings, click Local Policies, and then click Security Options.
3. In the right pane, view the Effective Settings column to verify that the correct security settings have been applied.

Note: As you will be applying security settings using Group Policy, it is very important you have a thorough understanding of their properties and interactions. The Microsoft white paper—Windows 2000 Group Policy, provides more detailed information on how they are deployed. For more details, see the "More Information" section at the end of this chapter.

Group Policy Structure

Group Policy configuration settings are stored in two locations:

- GPOs – located in Active Directory
- Security template files – located in the local file system

Changes made to the GPO are saved directly in Active Directory, whereas changes made to the security template files must then be imported back into the GPO within Active Directory before the changes can be applied.

Note: This operations book provides you with templates which can be used to modify your GPOs. If you make changes and modify the GPOs directly, they will be out of sync with the template files. You would therefore be advised to modify the template files and import them back into the GPO.

Security Template Format

Template files are text-based files. Changes to the template files can be made from the MMC snap-in Security Templates or by using a text editor such as Notepad. The following table shows how the policy sections maps to sections of the template files.

Table 8.1: Security Template Sections Corresponding to Group Policy Settings

Policy Section	Template Section
Account Policy	[System Access]
Audit Policy	[System Log] [Security Log] [Application Log]
User Rights	[Privilege Rights]
Security Options	[Registry Values]
Event Log	[Event Audit]
Restricted Groups	[Group Membership]
System Services	[Service General Setting]
Registry	[Registry Keys]
File System	[File Security]

Some sections within the security template file, such as the [File Security] and [Registry Keys], contain specific access control lists (ACLs). These ACLs are text strings, defined by the Security Descriptor Definition Language (SDDL). More information on editing security templates and on SDDL can be found on MSDN. For further details, see the “More Information” section at the end of this chapter.

Test Environment

It is vital that you thoroughly assess any changes to the security of your IT systems in a test environment before you make any changes to your production environment. Your test environment should mimic your production environment as closely as possible. At the very least, it should include multiple domain controllers and each member server role you will have in the production environment.

Testing is necessary to establish that your environment is still functional after you make changes, but is also vital to ensure that you have increased the level of security as intended. You should thoroughly validate all changes and perform vulnerability assessments on the test environment.

Note: Before anyone performs vulnerability assessments in your organization, you should ensure that they have obtained written permission to do so.

Checking your Domain Environment

Before implementing Group Policy in your production environment, it is important that the domain environment is stable and working properly. Some of the key areas in Active Directory that should be verified are DNS servers, domain controller replication, and time synchronization. You should also use a test environment to help ensure a stable production environment.

Verifying DNS Configuration

Name resolution by DNS is critical for servers and domain controllers to function properly. When multiple DNS servers are implemented for a domain, each DNS server should be tested. You should perform the following tests:

- On domain controllers:
  - Run dcdiag /v and netdiag /v using the verbose option to test DNS on each domain controller and review the output for any errors. DCDIAG and NETDIAG can be found on the Windows 2000 installation CD under the Support Tools directory.
  - Stop and start the Net Logon service and check the Event Log for any errors. The Net Logon service will dynamically register the service records in DNS for that domain controller and will produce error messages if it is not able to successfully register DNS records. These service records can be found in the file netlogon.dns located in the %SystemRoot%\System32\Config directory.
- On member servers, verify that DNS is operating correctly by using nslookup or running netdiag /v.

Domain Controller Replication

It is important that replication between multiple domain controllers is working properly before implementing Group Policy. If replication is not working correctly then changes made to Group Policy will not be applied to all domain controllers. This can create inconsistency between servers that are looking for Group Policy updates on domain controllers. Servers will be updated if they are pointing to the domain controller that the change was made on, while servers pointing to domain controllers that are still waiting for the Group Policy to be replicated will not be updated.

Forcing and Verifying Replication using Repadmin

Repadmin is a command-line tool included in the Support directory on the Windows 2000 CD. You can use repadmin to determine the directory replication partners of the destination server, and then issue a command to synchronize the source server with the destination server. This is done using the object globally unique identifier (GUID) of the source server.

► **To use repadmin to force replication between two domain controllers**

1. At a command prompt from a domain controller, type the following:
   repadmin /showreps <destination_server_name>
2. Under the Inbound Neighbors section of the output, find the directory partition that needs synchronization and locate the source server with which the destination is to be synchronized. Note the object GUID value of the source server.
3. Initiate replication by entering the following command:

   repadmin /sync

   <directory_partition_DN> <destination_server_name> <source_server_objectGuid>

Note: Once you have the object GUID of each domain controller, you could create a batch script that uses the repadmin tool to initiate replication between servers and provide status on whether the replication is successful.

Centralize Security Templates

It is very important that the security templates used for production are stored in a secure location that can only be accessed by the administrators responsible for implementing Group Policy. By default, security templates are stored in the %SystemRoot%\security\templates folder on each domain controller. This folder is not replicated across multiple domain controllers. Therefore you will need to select a domain controller to hold the master copy of the security templates so that you do not encounter version control problems with the templates.

Time Configuration

It is very important that system time is accurate and that all servers are using the same time source. The Windows 2000 W32Time service provides time synchronization for Windows 2000-based computers running in an Active Directory domain. The W32Time service ensures that Windows 2000-based clients' clocks are synchronized with the domain controllers in a domain. This is necessary for Kerberos authentication, but the time synchronization also assists in event log analysis.

The W32Time service synchronizes clocks using the Simple Network Time Protocol (SNTP) as described in RFC 1769. In a Windows 2000 forest, time is synchronized in the following manner:

- The primary domain controller (PDC) emulator operations master in the forest root domain is the authoritative time source for the organization.
- All PDC operations masters in other domains in the forest follow the hierarchy of domains when selecting a PDC emulator with which to synchronize their time.
- All domain controllers in a domain synchronize their time with the PDC emulator operations master in their domain as their in-bound time partner.
- All member servers and client desktop computers use the authenticating domain controller as their in-bound time partner.

To ensure that the time is accurate, the PDC emulator in the forest root domain should be synchronized to an external SNTP time server. You can configure this by running the following net time command, where <server_list> is your server list:

```
net time /setsntp:<server_list>
```

Note: If your PDC emulator in the forest root is behind a firewall, you may have to open UDP port 123 on the firewall to allow the PDC Emulator to connect to an Internet-based SNTP time server.

If your network uses older Windows operating systems, on these computers, clocks can be synchronized using the following command in a logon script where <timecomputer> is a domain controller on the network:

```
net time \\<timecomputer> /set /yes
```

Note: Computers running an operating system other than Windows should also synchronize their clocks to external time sources to allow logging events to be analyzed, based on time. For more information see the Microsoft Knowledge Base article Q216734,"How to Configure an Authoritative Time Server in Windows."

Policy Design and Implementation

If you are going to use Group Policy effectively, you must carefully determine how it will be applied. To simplify the process of applying and checking Group Policy security settings, we recommend that you apply security settings at two levels:

- **Domain Level**. To address the common security requirements, such as account policies and audit policies that must be enforced for all servers.
- **OU Level**. To address specific server security requirements that are not common to all the servers in the network. For example, the security requirements for infrastructure servers differ from those for servers running IIS.

Group Policy settings that affect security are divided into multiple sections.

Table 8.2: Sections of Group Policy and Their Purpose

Policy Section	Description
Account Policy\Password Policy	Password age, length and complexity configured
Account Policy\Account Lockout Policy	Lockout duration, threshold and reset counter configured
Account Policy\Kerberos Policy	Ticket lifetimes configured
Local Policies\Audit Policy	Enable/Disable recording of specific events
Local Policies\User Rights	Define rights such as log on locally, access from network and so on
Local Policies\Security Options	Modify specific security related registry values
Event Log	Success and Failure monitoring enabled
Restricted Groups	Administrators can control who belongs to a specific group
System Services	Controls Startup Mode for each service
Registry	Configure permissions on registry keys
File System	Configure permissions on folders, subfolders and files

All computers have a predefined local policy. When an Active Directory domain is initially created, default domain and domain controller policies are also created. Before you modify any default policies, it is important to document the settings they contain, so that you can easily return to the previous state in the event of a problem.

Server Roles

Exactly how much you can lock down a server will depend upon the role it is expected to perform in your environment. For example, a Domain Controller requires services to be started that a file and print server will not. There are potentially a large number of different roles that a Windows 2000 server may perform, but here

we concentrate on only two that will exist in an environment containing .NET-based applications.

Table 8.3: Windows 2000 Server Roles that support .NET-based applications.

Server Role	Description	Security Templates
Windows 2000 Domain Controller	An Active Directory domain controller	BaselineDC.inf
Windows 2000 Application Server	A locked down member server on which an application, such as Exchange 2000, can be installed. To allow the service to function correctly, security will have to be loosened.	Baseline.inf

The security requirements for each of these roles are different. Appropriate security settings for each role are discussed in detail in Chapter 9, "Securing Servers Based on Role." Then in Chapter 10, we introduce further templates and show how to use these to secure the specific servers running the .NET-based applications.

Note: This book assumes that servers perform specific defined roles. If your servers do not match these roles, or you have multipurpose servers, you should use the settings defined here as a guideline for creating your own security templates. However, you should keep in mind that the more functions each of your servers perform, the more vulnerable they are to attack.

Active Directory Structure to Support the Server Roles

As already mentioned, you can apply Group Policy in many different ways, using multiple GPOs, and at many different levels of hierarchy. For this book, we have defined a number of Group Policy settings that you can use to secure the various server roles. You will need to ensure that your Active Directory structure allows you to apply these settings.

To help you secure your Windows 2000-based environment, we have predefined some security templates that can be imported into GPOs. However, if you are going to use these as is, you will need to make sure that you have the appropriate Active Directory structure. The GPOs defined in this chapter are designed to be used with the OU structure shown in the diagram on the next page.

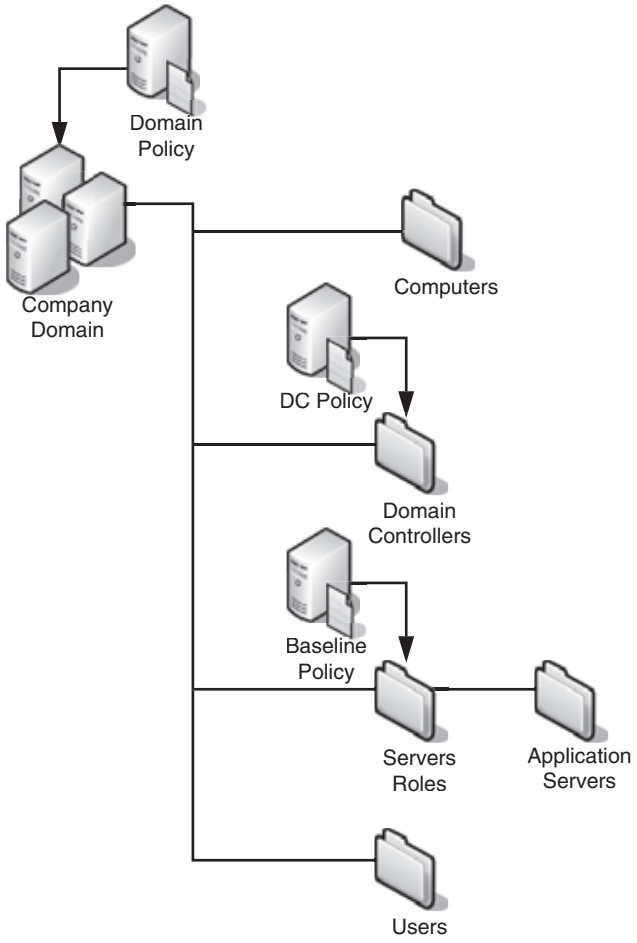


Figure 8.2
OU structure for use with defined GPOs

This OU structure can easily be extended to incorporate different server roles, for example you could create additional OUs for File and Print Servers under the Server Roles OU. In Chapter 10, we extend this OU structure further by creating OUs under the Application Servers OU.

Note: The domain structure is not important here, as domain and OU Group Policy only apply in the domain in which they were defined. The site structure is also unimportant as we do not define GPOs at the site level in this book.

► To create the OU structure

1. Start **Active Directory Users and Computers**.
2. Right-click the domain name, select **New**, and then select **Organizational Unit**.

3. Type **Member Servers** and then click **OK**.
4. Right-click **Member Servers**, select **New**, and then select **Organizational Unit**.
5. Type **Application Servers** and then click **OK**.

It is worth looking at the OU structure in some more detail.

Domain Level Policy

When a Windows 2000 domain is built, a default domain policy is created. For security settings that you want to apply to the whole domain, you can either:

- Create an additional policy and link it above the default policy
- Modify the existing default policy

Modifying the existing policy is generally simpler, however the advantage of creating an additional domain policy instead of modifying the default policy is that if there are problems with the additional policy, it can be disabled, leaving the default domain policy to resume control.

Remember that domains often contain client computers and users as well as servers. So if you are specifically looking to lock down servers, it will often be impractical to define the specific settings at the domain level. In practice, it is usually best to restrict your server security settings to those that must be set at the domain level.

In this operations book, we do not define specific settings at the domain level, as many, such as password length, will change according to the overall security policy of your organization. We do however make some general recommendations, which can be found in Chapter 9, "Securing Servers Based on Role."

Note: The password and account policy will ONLY affect domain accounts if they are set at the domain level (which means that you can only configure one password and account policy per domain). If these policies are set at the OU level or anywhere else, they will only affect local accounts. For more information, review the Knowledge Base article Q259576,"Group Policy Application Rules for Domain Controllers."

Member Servers OU

Many of the security settings you define for member servers should apply across every member server role. To simplify this process, we have created a baseline security template called Baseline.inf that you can import into a GPO and apply to the Member Servers OU. These settings will apply both to the Member Servers OU and any child OUs.

Domain Controllers OU

Windows 2000 already comes with a Domain Controllers OU. When a server becomes a domain controller, it is automatically placed here and you should not remove it, as it can cause user log on and access problems.

With this book, we provide you with a security template called BaselineDC.inf, that you can import into a GPO and apply to the Domain Controller OU. You may choose to apply this in addition to the Default Domain Controllers GPO, or simply modify the settings in the Default Domain Controllers GPO.

Individual Server Role OUs

The individual server role OUs are child OUs to the Member Server OU. This means that by default these servers will all take on the settings defined in your Member Server Baseline Policy.

If you use the baseline policy to secure your member servers, you will need to make alterations which will apply to each individual server role. You can do this by assigning GPOs to the individual server role OUs .

Importing the Security Templates

The following procedure imports the security templates included with this book into the OU structure suggested in this chapter. Before implementing the following procedure on a domain controller, you must extract the contents of the SecOps.exe file included with this book.

Warning: The security templates in this book are designed to increase security in your environment. It is quite possible that by installing the templates included with this book, you will lose some functionality in your environment. This could include the failure of mission critical applications. It is therefore ESSENTIAL that you thoroughly test these templates before deploying them in a production environment, and make any changes to them that are appropriate for your environment. Back up each domain controller and server prior to applying new security settings. Make sure the system state is included in the backup, because this is where the registry data is kept, and on domain controllers it also includes all of the objects in Active Directory.

► **Importing the Domain Controller Baseline Policy**

1. In **Active Directory Users and Computers**, right-click **Domain Controllers**, and then select **Properties**.
2. On the **Group Policy** tab, click **New** to add a new Group Policy object.
3. Type **BaselineDC Policy** and press **Enter**.
4. Right click **BaselineDC Policy** and select **No Override**.

   Note: This is required because the default domain controller policy configures all audit policy settings to No Auditing, with the exception of account management. Because the default domain controller policy has a higher precedence, the No Auditing setting will become the effective setting.

5. Click **Edit**.

6. Expand **Windows Settings**, right-click **Security Settings**, and select **Import Policy**.

   Note: If Import Policy does not appear on the menu, close the Group Policy window and repeat steps 4 and 5.

7. In the **Import Policy From** dialog box, navigate to **C:\SecurityOps\Templates**, and double-click **BaselineDC.inf**.
8. Close **Group Policy** and then click **Close**.
9. Force replication between your domain controllers so that all domain controllers have the policy.
10. Verify in Event Log that the policy was downloaded successfully and that the server can communicate with the other domain controllers in the domain.
11. Restart each domain controller one at a time to ensure that it reboots successfully.

► **Importing the Member Server Baseline Policy**

1. In **Active Directory Users and Computers**, right-click **Member Servers**, and then select **Properties**.
2. On the **Group Policy** tab, click **New** to add a new Group Policy object.
3. Type **Baseline Policy** and press **Enter**.
4. Click **Edit**.
5. Expand **Windows Settings**, right-click **Security Settings**, and select **Import Policy**.

   Note: If Import Policy does not appear on the menu, close the Group Policy window and repeat steps 4 and 5.

6. In the **Import Policy From** dialog box, navigate to **C:\SecurityOps\Templates**, and double-click **Baseline.inf**.
7. Close **Group Policy** and then click **Close**.
8. Force replication between your domain controllers so that all domain controllers have the policy.
9. Move a server for each role into the appropriate OU and on the server download the policy by using the **secedit** command.
10. Verify in Event Log that the policy was downloaded successfully and that the server can communicate with the domain controllers and with other servers in the domain. After successfully testing one server in the OU, move the remaining servers in the OU and then apply security.
11. Restart each server to ensure that they reboot successfully.

Keeping Group Policy Settings Secure

If you are applying security settings using Group Policy, it is important to ensure that the settings themselves are as secure as possible. This is generally achieved by ensuring that the permissions on both the GPOs and the OUs and domains on which they are applied are set appropriately. The templates included with this book do not modify the default Active Directory permissions, so you will need to modify these permissions manually.

Group Policy settings defined at higher level containers can potentially be overwritten by settings at lower level containers. Using the No Override option on the GPO prevents settings on a higher level container from being overwritten.

Note: Do not set **No Override** on the Member Server Baseline Policy. Doing so will prevent the server role policies from enabling the appropriate services and settings.

As well as separating the server roles at the OU level, you should also create separate corresponding administrator roles, assigning them administrative rights over only the corresponding OUs. This ensures that if an attacker manages to gain IIS server admin rights, they do not have access to infrastructure servers, and so on.

Only domain level administrators and above should have the rights to change the membership of an OU. If an OU level administrator can remove a server from that OU, they will be able to change the security settings on those servers.

Once policy has been applied to the servers, your work has not ended. You should check your servers on a regular basis to be sure that:

- The correct policy is applied to the server
- An administrator has not changed a setting in the policy and reduced the level of security on your servers
- Any policy updates or changes have been applied to all servers

Verifying that the settings in the GPO have been applied to your servers as expected will allow you to have confidence that your servers are properly secured. There are several methods that can be used to examine the Group Policy on a server in order to verify the policy is correctly set.

Events in the Event Log

If the policy is downloaded successfully, an Event Log event with the following information appears:

Type: Information

SourceID: SceCli

Event ID: 1704

Message String: Security policy in the Group Policy objects are applied successfully

It may take a few minutes for this message to appear after applying the policy. If you do not receive the successful Event Log message, you need to run **secedit /refreshpolicy machine_policy /enforce** and then restart the server to force the policy download. Check the Event Log again after the restart to verify the successful download of the policy.

Note: If services are set to Disabled in a GPO and the server is rebooted once, the services will typically have restarted before the settings defined in the GPO take effect. If you reboot the server a second time, this will ensure that the services set to Disabled are not started.

Verifying Policy Using Local Security Policy MMC

Another method for verifying that the policy has been applied successfully is to review the effective policy setting on the local server.

► **To verify the effective policy settings**

1. Start the **Local Security Policy** MMC.
2. Under **Security Settings**, click **Local Policies**, and then click **Security Options**.
3. In the right pane, view the **Effective Setting** column.

The Effective Setting column should display the settings that are configured in the template for the role of the selected server.

Verifying Policy Using Command Line Tools

There are also two command line tools that can be used to verify policy settings.

Secedit

This tool is included in Windows 2000 and can be used to display differences between the template file and the computer's policy. To compare a template with the current policy on a computer, use the following command line:

```
secedit /analyze /db secedit.sdb /cfg <template name>
```

Note: If you apply the templates included with this book and then run the above command, an access is denied error will be generated. This is expected due to the additional security applied. A log file will still be generated with the results of the analysis.

Gpresult

The *Windows 2000 Server Resource Kit* (Microsoft Press, ISBN: 1-57231-805-8) includes a tool called GPResult that can be used to display the policies currently applied to a server. To obtain a list of the policies applied to a server, use the following command line:

```
Gpresult /c
```

Note: Gpresult is covered in more detail in the "Troubleshooting Group Policy" section later in this chapter.

Auditing Group Policy

It is possible to audit changes to your Group Policy. Auditing policy changes can be used to keep track of who is changing, or attempting to change, policy settings. Auditing the success and failure of policy changes is enabled in the baseline security templates.

Troubleshooting Group Policy

Even though Group Policy is automatically applied, it is possible that the resulting Group Policy on a server is not as expected, mainly because Group Policy can be configured at multiple levels. This section provides some guidelines that can be used to troubleshoot Group Policy.

Note: If you are having a specific problem with Group Policy not covered in this chapter, be sure to check the Microsoft Knowledge Base. Some key Knowledge Base articles related to Group Policy are detailed in the "More Information" section at the end of this chapter as well as the "Troubleshooting Group Policy" whitepaper.

Resource Kit Tools

GPResult and GpoTool are two *Windows 2000 Server Resource Kit* tools that will help you troubleshoot Group Policy problems.

Note: These tools are also available online, see the "More Information" section at the end of this chapter for details.

GPResult

This tool provides a list of all the GPOs that have been applied to a computer, what domain controller the GPOs came from, and the date and time the GPOs were last applied.

When running GPResult on a server to ensure it has the correct GPOs, use the **/c** switch to display information on computer settings only.

When GPResult is used with the **/c** switch, it provides the following general information:

- Operating System
  - Type (Professional, Server, domain controller)

  - Build number and Service Pack details
  - Whether Terminal Services is installed and, if so, the mode it is using
- Computer Information
  - Computer name and location in Active Directory (if applicable)
  - Domain name and type (Windows NT or Windows 2000)
  - Site name

GPResult with the **/c** switch also provides the following information about Group Policy:

- The last time policy was applied and the domain controller that applied policy, for the user and computer
- The complete list of applied Group Policy objects and their details, including a summary of the extensions that each Group Policy object contains
- Registry settings that were applied and their details
- Folders that are redirected and their details
- Software management information detailing assigned and published applications
- Disk quota information
- IP Security settings
- Scripts

GpoTool

This command-line tool allows you to check the health of the Group Policy objects on domain controllers including:

- **Check Group Policy object consistency**. The tool reads mandatory and optional directory services properties (version, friendly name, extension GUIDs, and Windows 2000 system volume (SYSVOL) data (Gpt.ini)), compares directory services and SYSVOL version numbers, and performs other consistency checks. Functionality version must be 2 and user/computer version must be greater than 0 if the extensions property contains any GUID.
- **Check Group Policy object replication**. It reads the GPO instances from each domain controller and compares them (selected Group Policy container properties and full recursive compare for Group Policy template).
- **Display information about a particular GPO**. Information includes properties that cannot be accessed through the Group Policy snap-in such as functionality version and extension GUIDs.
- **Browse GPOs**. A command-line option can search policies based on friendly name or GUID. A partial match is also supported for both name and GUID.

- **Preferred domain controllers**. By default, all available domain controllers in the domain will be used; this can be overwritten with the supplied list of domain controllers from the command line.
- **Provide cross-domain support**. A command-line option is available for checking policies in different domains.
- **Run in verbose mode**. If all policies are fine, the tool displays a validation message; in case of errors, information about corrupted policies is printed. A command-line option can turn on verbose information about each policy being processed.

Use the following command line to obtain details of a Group Policy as well as if any errors in the policy are detected:

```
GPOTool /gpo:<gpo name>
```

Group Policy Event Log Errors

Some Group Policy Event Log errors indicate specific problems with your environment. Here are two that will prevent Group Policy from being properly applied:

- On a domain controller, warning event 1202 combined with error event 1000. This generally means that a domain controller has been moved from the Domain Controllers OU to another OU which does not have the Default Domain Controllers GPO linked.
- When an administrator attempts to open one of the default GPOs, the following error is returned:

  Failed to open Group Policy Object

  You may not have appropriate rights.

  Details: Unspecified Error

  In the event log, events 1000, 1001 and 1004 appear. This is due to a corrupt registry.pol file. By deleting the registry.pol file under SYSVOL, rebooting and making a change to the server, the errors should disappear.

Summary

Windows 2000 Group Policy is a very useful way to provide consistent settings across your Windows 2000-based environment. To deploy it effectively, you should ensure that you are aware of where GPOs are applied, that all of your servers are receiving the appropriate settings, and that you have defined appropriate security on the GPOs themselves.

More Information

Microsoft Whitepaper on Group Policy:

http://www.microsoft.com/windows2000/techinfo/howitworks/management/grouppolwp.asp

Microsoft Whitepaper on Troubleshooting Group Policy:

http://www.microsoft.com/Windows2000/techinfo/howitworks/management/gptshoot.asp

Knowledge Base articles on Group Policy Troubleshooting:

http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q250842

http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q216359

Administrative Template File Format:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/policy/policyref_17hw.asp

Security Descriptor Definition Language:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/accctrl_757p.asp

Additional tools and Group Policy information are available in:

The *Windows 2000 Server Resource Kit* (Microsoft Press, ISBN: 1-57231-805-8) or online at: *http://www.microsoft.com/windows2000/techinfo/reskit/default.asp*

9

Securing Servers Based on Role

Introduction

In the previous chapter we looked at how Group Policy can be used to define security settings on your servers. In this chapter we get into specifics, looking at baseline policies that can be defined for all member servers and domain controllers in the enterprise.

This approach allows administrators to lock down their servers using centralized baseline policies, applied consistently across all servers in the enterprise. The baseline policies allow only minimal functionality, but do allow servers to communicate with other computers in the domain and be authenticated against domain controllers. From this more secure state additional incremental policies can be applied, allowing each server to only perform the specific tasks defined by their role. Your risk management strategy will determine whether making these changes is appropriate for your environment.

This operations book partitions policy implementation in the following way:

- **Domain Wide Policy**. Address common security requirements, such as account policies that must be enforced for all servers and workstations.
- **Domain Controller Policy**. Policies that apply to the Domain Controllers OU. Specifically, the configuration settings impact audit policy, security options, and service configuration.
- **Member Server Baseline Policy**. Common settings for all member servers including audit policies, service configuration, policies that restrict access to the registry, file system, as well as other specific security settings, such as clearing the virtual memory page file on system shut down.
- **Server Role Policy**. Here we define a specific server role- that of the Application Server. Exactly how this role is defined depends on the particular applications that run in. We will expand this concept further in Chapter 10 “Securing Servers

Running .NET-based Applications" when we examine specific Application Server Roles.

This chapter deals with these policies and other settings that should be defined for particular server roles. For more information on how Group Policy is used to apply security settings, see Chapter 8, "Managing Security with Windows 2000 Group Policy."

Domain Policy

In this operations book, we do not enforce specific settings at the domain level, as many of these settings, such as password length, will change according to the overall security policy of your organization. It is, however, very important that you define these settings appropriately.

Password Policy

By default, a standard password policy is enforced for all servers in the domain. The table lists the settings for a standard password policy, and recommended minimums for your environment.

Table 9.1 Password Policy Default and Recommended Settings

Policy	Default Setting	Recommended Minimum Setting
Enforce password history	1 password remembered	24 passwords remembered
Maximum password age	42 days	42 days
Minimum password age	0 days	2 days
Minimum password length	0 characters	8 characters
Password must meet complexity requirements	Disabled	Enabled
Store password using reversible encryption for all users in the domain	Disabled	Disabled

Complexity Requirements

When the **Password must meet complexity requirements** setting of Group Policy is enabled, it requires passwords to be at least 6 characters in length (although we recommend you set this to 8 characters). It also requires that passwords contain characters from at least three of these classes:

- English upper case letters A, B, C, … Z
- English lower case letters a, b, c, … z
- Westernized Arabic numerals 0, 1, 2, … 9
- Nonalphanumeric characters such as punctuation symbols

Note: A password policy should not only be enforced on servers running Windows 2000, but also on any other devices requiring a password for authentication. Network devices, such as routers and switches, are very susceptible to attack if they are using simple passwords. Attackers may try to gain control of these network devices in order to bypass firewalls.

Account Lockout Policy

An effective account lockout policy will help prevent an attacker from successfully guessing the passwords of your accounts. The table lists the settings for a default account lockout policy and recommended minimums for your environment.

Table 9.2: Account Policy Default and Recommended Settings

Policy	Default Setting	Recommended Minimum Setting
Account Lockout Duration	Not Defined	30 minutes
Account Lockout Threshold	0	5 invalid logon attempts
Reset account lockout after	Not Defined	30 minutes

With the recommended minimums listed here, an account that has five invalid logon attempts within 30 minutes is locked out for 30 minutes (after which it will be reset back to 0 bad attempts and log on can be attempted again). The account can only be activated before the 30 minutes are up if an administrator resets the lockout. To increase the level of security in your organization, you should consider increasing the account lockout duration and decreasing the account lockout threshold.

Note: The password and account policy **must** be set at the domain level. If these policies are set on the OU level or anywhere else in Active Directory, they will affect local accounts and not domain accounts. It is only possible to have one domain account policy, for more information see Knowledge Base article Q255550,"Configuring Account Policies in Active Directory."

Member Server Baseline Policy

Once you have configured settings at the domain level, it is time to define common settings for all your member servers. This is done through a GPO at the Member Server OU, known as a baseline policy. A common GPO automates the process of configuring specific security settings on each server. You will also need to manually apply some additional security settings that cannot be done using group policies.

Baseline Group Policy for Member Servers

The configuration of the baseline policy used in this book is drawn from the hisecws.inf policy included with server and workstation installs. Some of the areas that hisecws.inf addresses include:

- **Audit Policy**. Determines how auditing is performed on your servers.
- **Security Options**. Determines specific security settings using registry values.
- **Registry Access Control Lists**. Determines who can access the registry.
- **File Access Control Lists**. Determines who can access the file system.
- **Service Configuration**. Determines which services are started, stopped, disabled, and so on.

For this book we have altered hisecws.inf to make it more secure. The Member Server Baseline Policy, baseline.inf, will help to create a server that is significantly more resistant to attack in production environments.

Hisecws.inf has been altered by adding:

- Registry values pertaining to security
- Service configuration
- Tighter file access control lists
- Enhanced auditing configuration

Member Server Baseline Auditing Policy

The settings for the application, security, and system event logs, are configured in the policy and applied to all member servers in the domain. The size for each of the logs is set at 10 megabyte (MB), and each log is configured to not overwrite events. Therefore, it is important for an administrator to regularly review and archive or clear the logs as appropriate.

Note: If a management system regularly monitors the logs for specific events, and extracts and forwards details to a management database, you will capture the necessary data and therefore can set the log files to overwrite.

The table shows the settings defined in the Member Server Baseline Auditing Policy.

Table 9.3: Member Server Baseline Audit Policy Settings

Policy	Computer Setting
Audit account logon events	Success, Failure
Audit account management	Success, Failure
Audit directory service access	Failure
Audit logon events	Success, Failure

Policy	Computer Setting
Audit object access	Success, Failure
Audit policy change	Success, Failure
Audit privilege use	Failure
Audit process tracking	No Auditing
Audit system events	Success, Failure
Restrict guest access to the application log	Enabled
Restrict guest access to the security log	Enabled
Restrict guest access to the system log	Enabled
Retention method for application log	Do not overwrite events (clear log manually)
Retention method for security log	Do not overwrite events (clear log manually)
Retention method for system log	Do not overwrite events (clear log manually)
Shut down the computer when the security audit log is full	Not Defined

Note: The retention method policy settings **Manually** is shown, which means do not overwrite events (clear log manually).

Member Server Baseline Security Options Policy

The following security options are configured in the baseline group policy.

Table 9.4: Member Server Baseline Security Options Policy Settings

Option	Setting
Additional restrictions for anonymous connections	No access without explicit anonymous permissions
Allow server operators to schedule tasks (domain controllers only)	Disabled
Allow system to be shut down without having to log on	Disabled
Allowed to eject removable NTFS media	Administrators
Amount of idle time required before disconnecting session	15 minutes
Audit the access of global system objects	Disabled
Audit use of Backup and Restore privilege	Disabled

(continued)

Option	Setting
Automatically log off users when logon time expires	Not Defined (see note)
Automatically log off users when logon time expires (local)	Enabled
Clear virtual memory page file when system shuts down	Enabled
Digitally sign client communication (always)	Enabled
Digitally sign client communication (when possible)	Enabled
Digitally sign server communication (always)	Enabled
Digitally sign server communication (when possible)	Enabled
Disable CTRL+ALT+DEL requirement for logon	Disabled
Do not display last user name in logon screen	Enabled
LAN Manager Authentication Level	Send NTLMv2 responses only, refuse LM & NTLM
Message text for users attempting to log on	
Message title for users attempting to log on	
Number of previous logons to cache (in case domain controller is not available)	0 logons
Prevent system maintenance of computer account password	Disabled
Prevent users from installing printer drivers	Enabled
Prompt user to change password before expiration	14 days
Recovery Console: Allow automatic administrative logon	Disabled
Recovery Console: Allow floppy copy and access to drives and folders	Disabled
Rename administrator account	Not defined
Rename guest account	Not defined
Restrict CD-ROM drive access to locally logged-on user only	Enabled
Restrict floppy access to locally logged-on user only	Enabled

Option	Setting
Secure channel: Digitally encrypt or sign secure channel data (always)	Enabled
Secure channel: Digitally encrypt secure channel data (when possible)	Enabled
Secure channel: Digitally sign secure channel data (when possible)	Enabled
Secure channel: Require strong (Windows 2000 or later) session key	Enabled
Secure system partition (for RISC platforms only)	Not defined
Send unencrypted password to connect to third-party SMB servers	Disabled
Shut down system immediately if unable to log security audits	Enabled (see the second note)
Smart card removal behavior	Lock Workstation
Strengthen default permissions of global system objects (for example, Symbolic Links)	Enabled
Unsigned driver installation behavior	Do not allow installation
Unsigned non-driver installation behavior	Warn but allow installation

Note: The default domain policy configures **Automatically log off users when logon time expires** to disabled. To configure this option you must edit the default domain policy and therefore it is not defined in the baseline policies included with this book.

Note: If you significantly increase the number of objects you audit, you run the risk of filling the security log and thus forcing a shutdown of the system. The system will then not be usable until an administrator clears the log. To prevent this, you should either disable the shutdown option listed in the table, or preferably, increase the security log size.

Some of the options set here need further discussion as they directly affect the way servers communicate with each other in the domain and can also have an impact on server performance.

Additional Restrictions for Anonymous Connections

By default, Windows 2000 allows anonymous users to perform certain activities such as enumerating the names of domain accounts and network shares. This allows an attacker to view these accounts and share names on a remote server without

having to authenticate with a user account. To better secure anonymous access, **No access without explicit anonymous permissions** can be configured. The effect of this is to remove the Everyone group from the anonymous users token. Any anonymous access to a server will not be allowed, and will require explicit access to any resources.

Note: For details on the effect this may have in your environment, see Knowledge Base article Q246261, "How to Use the RestrictAnonymous Registry Value in Windows 2000."

LAN Manager Authentication Level

The Microsoft Windows 9x and Windows NT® operating systems cannot use Kerberos for authentication, and so, by default, they use the NTLM protocol for network authentication in a Windows 2000 domain. You can enforce a more secure authentication protocol for Windows 9x and Windows NT by using NTLMv2. For the logon process, NTLMv2 introduces a secure channel to protect the authentication process.

Note: If you do use NTLMv2 for legacy clients and servers, Windows 2000-based clients and servers will continue to authenticate with Windows 2000 domain controllers using Kerberos. For information on enabling NTLMv2, see Knowledge Base article Q239869, "How to Enable NTLM 2 Authentication for Windows 95/98/2000/NT." Windows NT 4.0 requires service pack 4 to support NTLMv2 and Windows 9x platforms need the directory service client installed in order to support NTLMv2.

Clear Virtual Memory Page File when System Shuts Down

Important information kept in real memory may be dumped periodically to the page file. This helps Windows 2000 handle multitasking functions. If you enable this option, Windows 2000 clears the page file when the system is shut down, removing all information stored there Depending on the size of the page file, it could take several minutes before the system is completely shut down.

Digitally Sign Client/Server Communication

Implementing digital signing in high security networks helps to prevent impersonation of clients and servers (known as session hijacking or man in the middle attack). Server message block (SMB) signing authenticates both the user and the server hosting the data. If either side fails the authentication, data transmission will not take place. When SMB signing is implemented, there will be a performance overhead of up to 15 percent in order to sign and verify each packet between the servers. For more information on the performance overhead impact, see Knowledge Base article Q161372, "How to Enable SMB Signing in Windows NT."

Additional Security Options

For this book, additional registry values were added to the baseline security template file that are not defined within the Administrative Template (ADM) file. This means that when you load the MMC Security Templates snap-in and view the baseline.inf template, the registry values in tables 9.5 – 9.9 are not represented. Instead, these settings can be added to the .inf file using a text editor and will be applied to the server when the policy is downloaded.

Note: For more information on the relationship between .inf and .adm files, see Knowledge Base article Q228460, "Location of ADM (Administrative Template) Files in Windows."

These settings are embedded within the Baseline.inf security template in order to automate the changes. If the policy is removed, these settings are not automatically removed with it and must be manually changed.

Security Considerations for Network Attacks

Some denial of service attacks can pose a threat to the TCP/IP stack on Windows 2000-based servers. These registry settings help to increase the resistance of the Windows 2000 TCP/IP stack to standard types of denial of service network attacks. Information on these settings can be found in Knowledge Base article Q315669, "HOW TO: Harden the TCP/IP Stack in Windows 2000 Against Denial of Service."

The following registry keys have been added to the template file as subkeys of **HKLM\System\CurrentControlSet\Services\Tcpip|Parameters**:

Table 9.5: TCP/IP Parameters Added to the Registry by the Member Server Baseline Policy

Key	Format	Value (Decimal)
EnableICMPRedirect	DWORD	0
EnableSecurityFilters	DWORD	1
SynAttackProtect	DWORD	2
EnableDeadGWDetect	DWORD	0
EnablePMTUDiscovery	DWORD	0
KeepAliveTime	DWORD	300,000
DisableIPSourceRouting	DWORD	2
TcpMaxConnectResponseRetransmissions	DWORD	2
TcpMaxDataRetransmissions	DWORD	3
NoNameReleaseOnDemand	DWORD	1
PerformRouterDiscovery	DWORD	0
TCPMaxPortsExhausted	DWORD	5

Windows Sockets applications such as FTP servers and Web servers have their connection attempts handled by Afd.sys. Afd.sys has been modified to support large numbers of connections in the half open state without denying access to legitimate clients. This is accomplished by allowing the administrator to configure a dynamic backlog. The new version of Afd.sys supports four new registry parameters that can be used to control the dynamic backlog behavior. For more details on these settings, see Knowledge Base article Q142641, "Internet Server Unavailable Because of Malicious SYN Attacks."

The following registry keys have been added to the template file as subkeys of **HKLM\System\CurrentControlSet\Services\AFD\Parameters**:

Table 9.6: Afd.sys Settings Added to the Registry by the Member Server Baseline Policy

Key	Format	Value (Decimal)
DynamicBacklogGrowthDelta	DWORD	10
EnableDynamicBacklog	DWORD	1
MinimumDynamicBacklog	DWORD	20
MaximumDynamicBacklog	DWORD	20000

Disable Auto Generation of 8.3 Filenames

Windows 2000 supports 8.3 file name formats for backward compatibility with16-bit applications. This means that an attacker only needs 8 characters to refer to a file that may be 20 characters long. If you avoid using 16-bit applications you can turn this feature off. Disabling short name generation on an NTFS partition also increases directory enumeration performance.

The following registry key has been added to the template as a subkey of **HKLM\System\CurrentControlSet\Control\FileSystem**:

Table 9.7: Setting to Remove 8.3 Filename Creation Added to the Registry by the Member Server Baseline Policy

Key	Format	Value (Decimal)
NtfsDisable8dot3NameCreation	DWORD	1

Note: If you apply this setting to an existing server that already has files with auto generated 8.3 file names, it does not remove them. To remove existing 8.3 file names, you will need to copy those files off the server, delete the files from the original location, and then copy the files back to their original locations.

Disable Lmhash Creation

Windows 2000-based servers can authenticate computers running all previous versions of Windows. However, previous versions of Windows do not use Kerberos for authentication, so Windows 2000 supports Lan Manager (LM), Windows NT (NTLM) and NTLM version 2 (NTLMv2).The LM)hash is relatively weak compared to the NTLM hash and therefore prone to rapid brute force attack. If you do not have clients that require LM authentication you should disable the storage of LM hashes. Windows 2000 Service Pack 2 provides a registry setting to disable the storage of the LM hashes.

To disable the creation of the LMhash, you must manually add a NoLMHash key to the following registry key: **HKLM\SYSTEM\CurrentControlSet\Control\Lsa**:

Note: If you want to disable the creation of the LMhash, this key must be manually added to the registry. There is no way to use Group Policy to add a key to the registry, only values can be added.

For more information, see the Microsoft Knowledge Base article 299656, " New Registry Key to Remove LM Hashes from Active Directory and Security Account Manager."

Note: To disable the storage of LM hashes with this registry setting you must be running Windows 2000 Service Pack 2 or later.

Configuring NTLMSSP Security

The NTLM Security Support Provider (NTLMSSP) allows you to specify the minimum required security setting for server side network connections by applications.

The Member Server Baseline Policy ensures that the connection will fail if message confidentiality is in use but 128-bit encryption is not negotiated.

The following registry key has been added to the template as a subkey of **HKLM\SYSTEM\CurrentControlSet\Control\Lsa\MSV1_0**:

Table 9.8: Setting to configure NTLMSSP Security added to the registry by the Member Server Baseline Policy

Key	Format	Value (Hex)
NtlmMinServerSec	DWORD	0x20000000

Disabling Autorun

Autorun begins reading from a drive as soon as media is inserted in it. As a result, the setup file of programs and the sound on audio media starts immediately. To prevent a possible malicious program from starting when media is inserted the Group Policy disables Autorun on all drives.

The following registry key has been added to the template as a subkey of **HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer**:

Table 9.9: Setting to Disable Autorun on all Drives, Added to the Registry by the Member Server Baseline Policy

Key	Format	Value (Hex)
NoDriveTypeAutoRun	DWORD	0xFF

Member Server Baseline Registry Access Control Lists Policy

The Member Server Baseline Policy does not change the registry ACLs defined in hisecws.inf. You should perform careful testing in your environment before you make any changes.

The ACLs defined in hisecws.inf mainly change the Power Users group, which is created by default for backward compatibility with Windows NT 4.0–based environments. The template ensures that Power Users has the same permissions as the Users group on Windows 2000.

Note: The Power Users group is not defined on domain controllers.

Member Server Baseline File Access Control Lists Policy

To further secure the file system, you should ensure that more restrictive permissions are applied to directories and files common to all member servers in the domain. The Member Server Baseline Security Template incorporates all the file access control lists provided with the hisecws.inf template and adds settings for a number of folders and files.

Note: For details on the default registry and file permissions in Windows 2000, see the "Default Access Control Settings in Windows 2000" white paper available on TechNet. The "More Information" section at the end of this chapter has the link to the white paper.

The table shows the additional folders secured by the Member Server Baseline Policy in addition to those defined by the settings in hisecws.inf.

Table 9.10: Settings to Secure Key Directories Defined in the Member Server Baseline Policy

Folders Secured	Permissions Applied
%systemdrive%\	Administrators: Full control System: Full control Authenticated Users: Read and Execute, List Folder Contents, and Read

Folders Secured	Permissions Applied
%SystemRoot%\Repair %SystemRoot%\Security %SystemRoot%\Temp %SystemRoot%\system32\Config %SystemRoot%\system32\Logfiles	Administrators: Full control Creator/Owner: Full control System: Full control
%systemdrive%\Inetpub	Administrators: Full control System: Full control Everyone: Read and Execute, List Folder Contents, and Read

Note: %SystemRoot% defines the path and folder name where the Windows system files are located and %SystemDrive% defines the drive containing %systemroot%.

There are also a large number of files installed on the server that should be locked down further. The Member Server Baseline Policy will alter the ACLs on the default Windows startup files and also on many of the executables that can be run from the command prompt. The files affected are listed in Appendix A.

Member Server Baseline Services Policy

When Windows 2000 Server is first installed, default services are created and are configured to run when the system starts. Some of these services do not need to run in many environments, and as any service is a potential point of attack, you should disable unnecessary services.

The Member Server Baseline Policy only enables the services required for a Windows 2000 member server to participate in a Windows 2000 domain and provide basic management services.

Table 9.11: Services Enabled by the Member Server Baseline Policy

Service	Startup Type	Reason for inclusion in Member Server Baseline
COM+ Event Services	Manual	Allows management of Component Services
DHCP Client	Automatic	Required to update records in Dynamic DNS
Distributed Link Tracking Client	Automatic	Used to maintain links on NTFS volumes
DNS Client	Automatic	Allows resolution of DNS names
Event Log	Automatic	Allows event log messages to be viewed in Event log
Logical Disk Manager	Automatic	Required to ensure dynamic disk information is up to date

(continued)

Service	Startup Type	Reason for inclusion in Member Server Baseline
Logical Disk Manager Administrative Service	Manual	Required to perform disk administration
Netlogon	Automatic	Required for domain participation
Network Connections	Manual	Required for network communication
Performance Logs and Alerts to log or triggers alerts	Manual	Collects performance data for the computer, writes it
Plug and Play system hardware	Automatic	Required for Windows 2000 to identify and use
Protected Storage keys	Automatic	Required to protect sensitive data such as private
Remote Procedure Call (RPC)	Automatic	Required for internal processes in Windows 2000
Remote Registry Service	Automatic	Required for MBSA utility (see Note)
Security Accounts Manager	Automatic	Stores account information for local security accounts
Server	Automatic	Required for MBSA utility
System Event Notification	Automatic	Required to record entries in the event logs
TCP/IP NetBIOS Helper Service	Automatic	Required for software distribution in Group Policy (may be used to distribute patches)
Windows Management Instrumentation Driver	Manual	Required to implement performance alerts, using Performance Logs and Alerts
Windows Time	Automatic	Required for Kerberos authentication to consistently function
Workstation	Automatic	Required to participate in a domain

These settings assume a pure and standard Windows 2000-based environment (with the exception of the MBSA tool). If your environment includes Windows NT 4.0 (or you have other tools on all your member servers) you may require other services for compatibility purposes. If you do enable other services, these may in turn have dependencies that require further services. Services needed for a specific server role can be added in the policy for that server role.

Appendix B shows all the services present in a default installation of Windows 2000 and Appendix C shows the additional services that may be added to a default installation.

Key Services Not Included in the Member Server Baseline

The goal of the Member Server Baseline Policy is to be as restrictive as possible. For this reason several services are disabled that may be required in your environment. Some of the more common ones are listed here.

SNMP Service

In many cases, management applications require an agent to be installed on each server. Typically, these agents will use SNMP to forward alerts back to a centralized management server. If management agents are required then you should check to see if they need the SNMP service started.

WMI Services

The Windows Management Instrumentation (WMI) service is disabled in the Member Server Baseline Policy. To manage logical disks using computer management, you need to enable the WMI service. Many other applications and tools also use WMI.

Messenger Service and Alert Service

Although not explicitly dependent on one another, these services work together to send administrative alerts. The Messenger service will send alerts triggered by the Alert service. If you are using Performance Logs and Alerts to trigger alerts, you will need to enable these services.

Domain Controller Baseline Policy

All domain controllers created in the domain are automatically assigned to the Domain Controllers OU. Domain controllers should never be moved out of the Domain Controllers OU as there are specific security ACLs applied to this OU.

The Domain Controllers OU is a top level OU and so will not take on the settings defined in your Member Server Baseline Policy. For this reason, we have created a separate Domain Controller Baseline Policy.

Configuration settings implemented in the Domain Controller Baseline Policy affects the following sections of the policy:

- Audit Policy
- Security Options
- Service Configuration

Note: File ACLs, with the exception of the System32 files listed in Appendix A, and registry ACLs are not included in this Group Policy, as they are defined and implemented when the server running Windows 2000 is promoted to a domain controller. A security template called Defltdc.inf is applied during the promotion of a Windows 2000-based server to a domain controller. This template applies ACLs to the file system and registry keys for the additional services created to support a domain controller.

Domain Controller Baseline Audit and Security Options Policy

The audit policy and security options configured for the domain controllers are identical to the baseline policy (see the "Member Server Baseline Policy" section for details on these settings.)

Domain Controller Baseline Services Policy

The services configured for startup are those defined in the member server baseline configuration, plus additional services needed to support the domain controller functions.

Table 9.12: Services Enabled by the Domain Controller Baseline Services Policy, in Addition to Those Set by the Member Server Baseline Policy

Service	Startup Type	Reason for inclusion in Domain Controller Baseline
Distributed File System	Automatic	Required for Active Directory Sysvol share
DNS Server	Automatic	Required for Active Directory integrated DNS
File Replication	Automatic	Needed for file replication between domain controllers
Kerberos Key Distribution Center	Automatic	Allows users to log onto the network using Kerberos v5
NT LM Security Support Provider	Automatic	Allows clients to log on using NTLM authentication
RPC Locator	Automatic	Allows the domain controller to provide RPC name service

Key Services Not Included in the Domain Controller Baseline Policy

The goal of the Domain Controller Baseline Policy is to be as restrictive as possible. For this reason several services are disabled that may be required in your environment. Some of the more common ones you may require are listed here.

Simple Mail Transport Protocol (SMTP)

Intersite replication can occur using either RPC or SMTP. If you use SMTP for replication in your environment, you will need to enable the SMTP Service.

Intersite Messaging

This service is used for mail-based replication between sites. Each transport to be used for replication is defined in a separate add-in dynamic link library (DLL). These add-in DLLs are loaded into Intersite Messaging. Intersite Messaging directs send requests and receive requests to the appropriate transport add-in DLLs, which

then route the messages to Intersite Messaging on the destination computer. If you use SMTP for replication in your environment, you will need to enable this service.

IIS Admin Service

If the SMTP service is started then the IIS Admin service also needs to be started as the SMTP service is dependent on the IIS Admin service.

Distributed Link Tracking Server Service

This service is used to track files on NTFS volumes throughout a domain and is contacted by computers running the Distributed Link Tracking Client service. These computers will periodically continue to attempt to contact the Distributed Link Tracking Server service even after it is disabled.

Note: If you run the dcdiag utility from the Windows 2000 Support Tools, it will check for all services which normally run on domain controllers to be started. As some services are disabled in the Domain Controller Baseline Policy, dcdiag will report errors. This is to be expected and does not indicate a problem with your configuration.

Other Baseline Security Tasks

It is not possible to perform all the tasks required to increase the security of your member servers and domain controllers using Group Policy. There are a number of additional steps you should take to increase the overall level of security on all of your servers.

Securing Built-in Accounts

Windows 2000 has a number of built-in user accounts, which cannot be deleted, but can be renamed. Two of the most commonly known built-in accounts on Windows 2000 are Guest and Administrator. By default, the Guest account is disabled on member servers and domain controllers. You should not change this setting. The built-in Administrator account should be renamed and the description altered to prevent attackers from compromising a remote server using a well known name. Many malicious scripts use the built-in administrator account as a first attempt for comprising the server.

Note: The built-in administrator account can be renamed using Group Policy. We have not implemented this setting in the baseline policies because you should choose a name which is not well known.

Securing Local Administrator Account

Every member server has a local accounts database and a local administrator account that provides full control over the server. This account is therefore very important. You should rename this account, and ensure that it has a complex password. You should also ensure that local administrator passwords are not replicated across member servers. If they are, an attacker who gains access to one member server will be able to gain access to all others with the same password.

You should not make local administrator accounts part of the Domain Admins group as this extends their capabilities beyond what is necessary to administer member servers. For the same reason, it is important to ensure that only local accounts are used to administer your member servers.

Securing Service Accounts

Windows 2000 services typically run under the Local System account, but they can also be run under a domain user or local account. You should use local accounts whenever possible over domain user accounts. A service runs under the security context of its service account, so if an attacker compromises a service on a member server, the service account can potentially be used to attack a domain controller. When determining which account to use as a service account, you should make sure that the assigned privileges are limited to what is required for the successful operation of the service. The table below explains the privileges inherent to each type of service account.

Table 9.13: Privileges of Windows 2000 Accounts in Different Environments

Authentication when running service on Windows 2000-based computers	Intraforest only, all Windows 2000-based servers	Multiforest application with NTLM trusts between domains
Local user service account	No network resources, local access only under account's assigned privileges	No network resources, local access only under account's assigned privileges
Domain user service account	Network access as domain user, local access under user's privileges	Network access as domain user, local access under user's privileges
LocalSystem	Network access as machine account authenticated user, local access under LocalSystem	No network resources spanning forests, local access under LocalSystem

All Windows 2000 default services run under **LocalSystem** and you should not change this. Any additional services added to the system requiring the use of domain accounts should be evaluated carefully before they are deployed.

Validating the Baseline Configuration

After security has been applied for the first time to a server, it is good practice to validate that the specific security settings have been configured correctly. The Microsoft Security Baseline Analyzer Tool will perform a series of tests against your servers, and warn you of any security problems you may encounter.

Validate Port Configuration

It is important to validate the final port configuration and to understand which TCP and UDP ports your servers running Windows 2000 are listening on. After applying the baseline policies, the netstat command can be run to show what ports the server is still listening on for each network interface card. The table shows the expected output netstat for a member server with the Member Server Baseline Policy applied:

Table 9.14: Ports a Member Server Will Listen on After the Member Server Baseline Policy is Applied

Protocol	Local Address	Foreign Address	Status
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
TCP	<IP Address>:139	0.0.0.0:0	LISTENING
UDP	<IP Address>:137	*.*	N/A
UDP	<IP Address>:138	*.*	N/A
UDP	0.0.0.0:445	*.*	N/A
UDP	0.0.0.0:1027	*.*	N/A
UDP	0.0.0.0:1045	*.*	N/A

Windows 2000 Application Server Role

Once you have applied your baseline policies, your servers will be significantly more secure. From this state, you may need to enable additional settings, adding functionality to your baseline.

The Windows 2000 Application Server is the most secure and locked down of the server roles. The goal of the secure application server role is to provide a very locked down server on which you can install an application, such as Exchange or SQL. This server role is designed so that all it can do is communicate with domain controllers for authentication purposes.

Note: It is possible to modify the templates included with this book to build templates for other roles. If this is done, it is important to fully test the modified template to ensure it provides the level of security desired.

Settings for the Application Server role will depend on the particular application you are deploying. For this reason, the settings are unchanged from the member server baseline. Therefore the application server role is very restricted—to install and run certain applications you will need to alter the security settings from the defaults defined here. The easiest way to accomplish this is to create a new OU for the application under the Application Servers OU. Then create a Group Policy that modifies the baseline settings and import the policy into the new OU. We use this method as the basis for locking down servers running .NET-based applications in the next chapter.

Changes to the Recommended Environment

The goal of the recommendations in this chapter is to create a significantly more secure environment for Windows 2000-based servers. However, some of the changes may not be appropriate for your organization. Here we look at two cases where 1) more administrative capability is required, and 2) where MBSA will not be used.

Administration Changes

The default baseline polices for member servers and domain controllers will eliminate some of the remote (and some of the local) administrative functionality from your environment. Remote management using the Microsoft Management Console (MMC) computer management snap-in will not work with the default baseline policies because some MMC related services are disabled.

The baseline policies enable the Server service and Remote Registry service. This will allow the computer management snap-in to remotely connect to other computers and administer these elements:

- Shared Folders
- Local users and groups
- Under Storage Management, everything except Logical Drives and Removable Storage
- Services Device Manager
- Event Viewer
- Performance Logs and Alerts

WMI is not enabled in the baseline policies. This prevents these elements from being administered:

- WMI
- Under Storage Management, Logical Drives

If you need to administer these, locally or remotely, you should enable the WMI service.

Removable Storage cannot be accessed remotely with just the Member Server Baseline Policy services started. If the Removable Storage service is not started on the remote server then the remote server will produce a DCOM error message in the event log stating that the service is not available.

Note: When enabling the above services to allow administration, enable the services only in the incremental server role policies that require the services.

Note: Some administration tools may require you to make security modifications on the client from which you are running the tool. For example, some tools may use NTLM authentication and the baseline policy configures the servers to only accept NTLM v2. See the "LAN Manager Authentication Level" section in this chapter for more information on configuring this.

Security Modifications if MBSA is Not Implemented

MBSA is a tool which allows you to verify which patches are installed on each of the servers in your organization. We highly recommend that you use a tool such as MBSA as it will help you increase the overall level of security in your environment.

However, if you do not implement MBSA, you can disable the Remote Registry service and Server service in the Member Server Baseline Policy. In the Domain Controller Baseline policy you can disable the Remote Registry service.

If you do disable the Server and Remote Registry services, you will also lose almost all of your remote administration capabilities.

Summary

Windows 2000-based servers provide a great deal of functionality out of the box. However, much of this functionality is not required for all servers. By defining the tasks that your servers perform, you can disable those elements you do not require, and therefore increase the security in your environment. If you implement the steps suggested in this chapter, you will go a long way toward making your environment significantly more secure.

More Information

Information on securing the Windows 2000 TCP/IP Stack:

http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/security/website/dosrv.asp

Default Access Control Settings in Windows 2000 white paper:

http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/prodtechnol/windows2000serv/maintain/featusability/secdefs.asp

Microsoft Security Toolkit:

http://www.microsoft.com/security/mstpp.asp

Glossary of Windows 2000 Services:

http://www.microsoft.com/windows2000/techinfo/howitworks/management/w2kservices.asp

10

Securing Servers Running .NET-Based Applications

Introduction

As you examine the security of your .NET Framework environment, you need to consider the security of the individual servers running .NET-based applications. Ensuring the security of Windows 2000 is fundamental to the security of the servers running the .NET-based applications, since these currently run in a Windows 2000-based environment. In Chapters 8 and 9 of this book, we showed a general strategy for securing servers running Windows 2000; using Group Policy templates and other settings to provide a locked-down state, then gradually modifying settings to ensure the functionality you require. In this chapter we extend that concept to include servers running .NET-based applications.

A number of different servers may be involved in providing the functionality of .NET-based applications. These servers will perform different roles and therefore will require different security settings.

In this section we will examine the different server types that would be used to run a typical distributed .NET-based application and discuss the security settings that would be appropriate for each type.

Test Environment

It is vital that you thoroughly assess any changes to the security of your IT systems in a test environment before you make any changes to your production environment. Your test environment should mimic your production environment as closely as possible. At the very least, it should include multiple domain controllers and each member server role you will have in the production environment.

Testing is necessary to establish that your environment is still functional after you make changes, but is also vital to ensure that you have increased the level of security as intended. You should thoroughly validate all changes and perform vulnerability assessments on the test environment.

Note: Before anyone performs vulnerability assessments in your organization, you should ensure that they have obtained written permission to do so.

Securing Server Roles for .NET-based Applications

We have supplied security templates with this book to modify the security on servers running the .NET Framework. You will need to import these templates in order for them to be applied to servers running the .NET Framework.

The following table defines the server roles and the templates used to increase their security.

Table 10.1: .NET-Based Application server roles

Server Role	Description	Security Templates
.NET Presentation Tier Server	Production server for presentation tier .NET-based applications	NETAppPresentation.inf
.NET Business Tier Server	Production server for Business tier .NET-based applications	Baseline.inf and NETAppBusiness.inf
.NET Presentation Tier Server (Development)	Server used for testing .NET-based applications still in development	Baseline.inf and NETAppPresentationDev.inf
.NET Business Tier Server (Development)	Server used for testing .NET-based applications still in development	Baseline.inf and NETAppBusinessDev.inf

Presentation Tier Server Role

Presentation tier servers are web servers that proxy requests from the client. For applications that you want to make available on the Internet, these servers should be physically separate from the corporate network, and protected from the Internet by a firewall. Also, they should not be part of your main corporate Active Directory structure. The settings on these servers will be similar to the IIS Server settings specified in Chapter 4, "Securing Servers by Role" in *Security Operations for Windows 2000*, but fewer services are required as the servers do not participate in a Windows 2000 domain.

Applying Security Template to Presentation Tier Servers

For this book, we have created a special template for Presentation tier servers, NETPresentation.inf. How this template should be applied will depend on whether your Presentation tier servers are part of an Active Directory forest or not. For the best security, you should not make these servers part of the corporate Active Directory forest. However, you have two alternatives. The first is to have these servers as part of a workgroup, the second is to create an entirely separate forest to manage these servers. If you have a small number of Presentation tier servers, using a workgroup is the best option because the overhead of manually applying Group Policy templates will be minimal and you will also not require additional hardware to support a separate forest. If you have a large number of Presentation tier servers, you should create a separate forest to make the application of Group Policy easy to manage.

The NETPresentation.inf file is a combination of the Member Server Baseline Policy and the IIS Server Role Policy from *Security Operations for Microsoft Windows 2000 Server,* with some customizations for the .NET Framework.

Note: The Member Server Baseline Policy and the IIS Server Role Policy are explained in detail in Chapter 4, "Securing Servers by Role" in *Security Operations for Microsoft Windows 2000 Server*.

The security templates are contained in the SecurityOps.exe file included with the book. You will need to extract this file prior to importing the security templates. Prior to installing the templates included with this book, you should ensure that you are running at least Windows 2000 with Service Pack 3 installed.

Warning: The security templates in this book are designed to increase security in your environment. It is quite possible that by installing the templates included with this book, you will lose functionality in your environment. This could include the failure of mission critical applications. Not only that, but there is no easy way to return to all of your previous settings once they have been applied. It is therefore ESSENTIAL that you thoroughly test these templates before deploying them in a production environment, and make any changes to them that are appropriate for your environment. Back up each domain controller and server prior to applying new security settings. Make sure the system state is included in the backup, because this is where the registry data is kept, and on domain controllers it also includes all of the objects in Active Directory.

If you do not make the Presentation tier servers part of a domain, Group Policy cannot be applied automatically from domain controllers to manage them. Instead, you should apply the security templates using the secedit command, or through the MMC security snap-in.

► **To apply the security template to the Presentation tier servers**

1. Open the command prompt and type:

```
secedit /configure /DB c:\presentation.sdb /CFG
c:\securityops\templates\NETAppPresentation.inf /overwrite /log
c:\presentation.txt /verbose
```

2. The following message appears:

   "To configure the current system with this template in the /overwrite mode will result in losing the existing security records in the database specified. Do you want to continue this operation ? [y/n]"

3. Type **y** to continue.
4. After the security configuration is complete, review the c:\presentation.txt log file to confirm that changes have been applied successfully.

Note: Some errors are to be expected in the log file. You will typically see "Warning 5: Access is denied" errors for several registry keys related to disk and display drivers. In addition, depending on the components and services installed on the server, you may also see "Warning 2: The system cannot find the file specified" for some files and "Error 1060: The specified service does not exist as an installed service" for some services. These errors are to be expected, as the security templates do not assume a default environment as a starting point.

---

5. You will then need to restart the computer and test the application to ensure that everything is working correctly. Check the event logs.

Note: See Chapter 8, "Managing Security with Windows 2000 Group Policy" for information on troubleshooting Group Policy.

Presentation Tier Services Policy

This policy combines the settings specified in the Windows 2000 Member Server Baseline Policy with the IIS Server Role Policy, and includes customizations for.NET-based applications. Further services are disabled as we are assuming that the servers are not participating in a domain.

Note: If a setting is not specified in this section, it is unchanged from the Windows 2000 Member Server Baseline Policy setting as defined in Chapter 9, "Securing Servers Based on Role."

Table 10.2: Shows the services specified in the Presentation Tier Server Policy

Service	Startup Type	Reason for Inclusion in Presentation Tier Server Policy
ASP.NET State Service	Manual	Required to support out-of-process session states for ASP.NET
IISAdmin	Automatic	Administration of the Web server
World Wide Web Publishing Service	Automatic	Provides Web server functionality
Machine Debug Manager	Disabled	Not required on a production server
DNS Client	Disabled	Not required as machines do not participate in a domain
DHCP Client	Disabled	Not required as machines do not participate in a domain
TCP/IP NetBIOS Helper	Disabled	Not required as machines do not participate in a domain
Net Logon	Disabled	Not required as machines do not participate in a domain

It is important to note that with the services specified above, the Presentation tier servers will not be able to perform either NetBIOS or DNS name resolution. In most cases this will not be a problem because you can always specify TCP/IP addresses directly. However, if you do require some host name resolution, you can implement a hosts file on the Presentation tier servers which specifies the server names you need to resolve.

Presentation Tier Auditing Policy

The settings here are the same as for the Member Server Baseline Policy, defined in Chapter 9, "Securing Servers Based on Role." However, there are circumstances under which you may want to change the auditing settings defined here. If you have an Internet facing application with extremely large numbers of logons being processed, it may become impractical to audit for successful logons, since the event log will rapidly fill. Under those circumstances you may wish to change the audit settings to process logon and account logon failures, but not successes. Under these circumstances you would also need to change the settings at the Domain Controller OU and possibly also for the Business Tier Servers OU.

Presentation Tier Account Lockout Policy

IUSR_COMPUTERNAME is used for anonymous access to IIS and IWAM_COMPUTERNAME is used to start out-of-process applications. One possible denial-of-service attack is to lock out either of these accounts. This could be achieved

relatively easily if a user attempts to authenticate against the server using either account. Therefore the Presentation tier does not configure Account Lockout settings.

If you decide to make Presentation tier servers part of their own forest, you should ensure that the web servers themselves are not domain controllers. As well as being good practice generally, local accounts will be used for IIS. This means a) if an account is compromised it will not affect more than one machine, and b) that you can set account lockout settings on the domain without affecting the IUSR and IWAM accounts on the web servers.

IUSR_COMPUTERNAME is a well-known account, and therefore more prone to attack. You should consider disabling the account, creating a new account, and configuring IIS to use it instead. Make sure you do not prefix the new account with IUSR. Doing so will cause conflicts with IIS 5.

Note: If you create a new account, you must ensure it has the appropriate user rights and permissions. The following Knowledge Base articles can help you determine these. Q323640, "HOW TO: Secure the IUSER_<Computer_name> Account" and Q300432, "HOWTO: Promote a Member Server Running IIS to a DC Running IIS."

Presentation Tier File Access Control Lists Policy

The .NET Framework just-in-time (JIT) compiler requires access to the Temp folder to store the temporary files created during compilation. To allow this, the Presentation Tier Server Policy modifies permissions on the Temp Folder:

Table 10.3: File permissions configured by the Presentation Tier Server Policy

Folder	Permissions Required
"%SystemRoot%\Temp	Administrators: Full Control Creator Owner: Full Control System: Full Control Users: Modify, Read and Execute, List Folder Contents, Read, and Write

Presentation Tier Additional Security Options

The Member Server Baseline Policy configures the **Tcp\Ip SynAttackProtect** registry parameter to **2**. This setting delays notifications to WinSock/AFD until the TCP three-way handshake is complete, and is therefore too restrictive for a Web server. The Presentation Tier Server Policy configures **SynAttackProtect** to **1**, which reduces the number of retries and delays the creation of a route cache entry (RCE).

Removing NetBIOS

The Presentation tier servers do not participate in the domain, so NetBIOS can be disabled on them. However, if you simply stop the services that use NetBIOS, the server will continue to listen for NetBIOS traffic on port 445. This port is used by all servers in the perimeter network (also known as DMZ) during the logon process to the internal domain controllers. To completely remove NetBIOS so the server will not listen on TCP port 445, disable the NetBIOS driver.

► **To disable the NetBIOS driver**

1. On each Presentation tier server, right-click **My Computer** and then click **Manage**.
2. Right-click **Device Manager,** point to **View**, and then click **Show Hidden Devices**.
3. Expand **Non Plug and Play Drivers**.
4. Right-click **NetBIOS over TCP/IP**, and then click **disable**.
5. Click **Yes**.
6. Click **Yes** to restart the computer.
7. After the computer has restarted, check that it is not listening to TCP 445 by running the following command from a command prompt:

```
netstat -an -p TCP
```

Note: The security template disables the TCP/IP NetBIOS Helper service and the DHCP Client service on the Presentation tier servers. If you have not disabled these services and you remove the NetBIOS driver, you will need to disable these services or you will receive an error from the service control manager when the server is restarted.

Business Tier Server Role

Unlike the Presentation tier servers, your Business tier servers should form part of either your corporate Active Directory forest, or a data center Active Directory forest if you want to isolate your data center from your corporate forest. In Chapter 8, "Managing Security with Windows 2000 Group Policy", we recommend an OU structure that allows you to easily adopt the security templates supplied and is extensible for additional server roles. In this chapter, the OU structure is extended to incorporate the Business Tier Policy.

To incorporate the new servers, we create a Framework Servers OU under the Application Servers OU and add further OUs for the Business tier under the Framework Servers OU.

The diagram below shows the OU structure recommended to accommodate the Business tier server role:

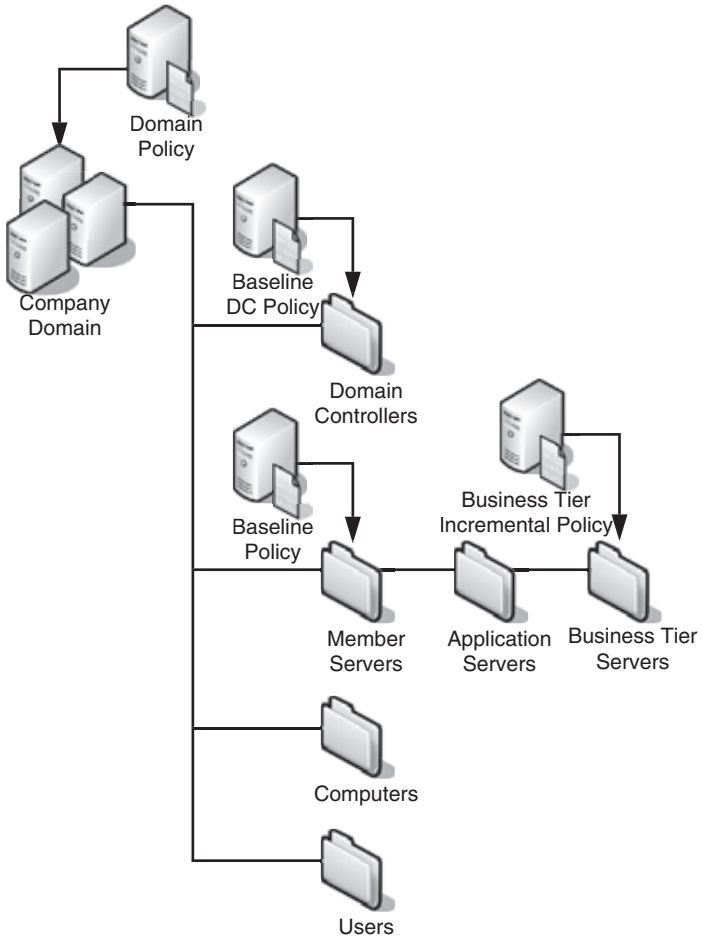


Figure 10.1
OU Structure with the Business Tier Servers OU added

Importing the Security Template

The following procedure imports the security templates included with this book into the OU structure suggested in this chapter.

► **To create the .NET Framework Server Group Policy objects and import the security templates**

1. Double click the **SecOps.exe** included with this book to extract to the templates required for this procedure.

2. In **Active Directory Users and Computers**, expand **Member Servers**, expand **Application Servers**, right-click **Business Tier Servers**, and then select **Properties**.
3. On the **Group Policy** tab, click **New** to add a new Group Policy object.
4. Type **Business Tier Incremental Policy** and press **Enter**.
5. Click **Edit**.
6. Under **Computer Configuration**, expand **Windows Settings**, right-click **Security Settings**, and select **Import Policy**.

   Note: If Import Policy does not appear on the menu, close the Group Policy window and repeat steps 4 and 5.

7. In the **Import Policy From** dialog box, navigate to **C:\SecurityOps\Templates**, and double-click **NETAppBusiness.inf**.
8. Close **Group Policy**, and then click **OK**.
9. Force replication between your domain controllers so that all domain controllers have the policy.
10. Move a server into the Business Tier Servers OU.
11. On the Business tier server that was moved to the Business Tier Servers OU download the policy by using the **secedit /refreshpolicy machine_policy /enforce** command.
12. Verify in event log that the policy was downloaded successfully and that the server can communicate with the domain controllers and with other servers in the domain. After successfully testing one server in the OU, move the remaining servers into the OU and then apply security.

    Note: For more information on verifying the success of the Group Policy download, see Chapter 8, "Managing Security with Windows 2000 Group Policy."

13. Repeat steps 10, 11, and 12 on all Business tier servers.
14. Restart each server to ensure that they reboot successfully.

Business Tier Server Incremental Policy

The Business Tier Server Incremental Policy defines additional settings to those defined in the Member Server Baseline Policy shown in Chapter 9, "Securing Servers Based on Role." These settings are designed to add minimal new functionality, allowing .NET-based applications to run. It may be necessary to modify them to get specific applications to run.

Note: If a setting is not specified in this section, it is unchanged from the Windows 2000 Member Server Baseline Policy setting as defined in Chapter 9, "Securing Servers Based on Role."

Business Tier Server Services Settings

The Business Tier Server Incremental Policy adds functionality to the standard Member Server Baseline Policy to allow the .NET-based applications to run.

Table 10.4: Service settings modified in the Business Tier Server Incremental Policy

Service	Startup Type	Reason for inclusion in Business Logic Server Incremental Policy
ASP.NET State Service	Manual	Required to support out-of-process session states for ASP.NET
IIS Admin Service	Automatic	Administration of the Web server
World Wide Web Publishing Service	Automatic	Provides Web server functionality
Machine Debug Manager	Disabled	Not required on a production server

Additional Security Options

The Member Server Baseline Policy configures the **Tcp\Ip SynAttackProtect** registry parameter to **2**. This setting is too restrictive on a Web server and may result in an unacceptable loss of performance. Therefore, the Presentation Tier Server Policy configures **SynAttackProtect** to **1**.

File Access Control Lists Policy

The .NET Framework JIT compiler requires access to the temp folder in order to store the temporary files created during compilation. Therefore, the Business Tier Server Incremental Policy configures the following file permissions:

Table 10.5: File Permissions Modified by the Business Tier Incremental Policy

Folder	Permissions Required
"%SystemRoot%\Temp	Administrators: Full Control Creator Owner: Full Control System: Full Control Users: Modify, Read and Execute, List Folder Contents, Read, and Write

Development Server Roles

As new applications are developed, it is extremely important that they are tested in a secure environment to ensure that a) they function properly in that environment and b) they do not compromise security in any way.

To ensure that applications in development can be tested properly, you should create a separate Active Directory forest, which maps closely to the forest in which the application will eventually live.

We have provided security templates to replace the production Presentation tier and Business tier templates in an application test environment. These templates allow the application to be debugged by configuring the Machine Debug Manager service to start automatically. There are no other differences between these templates and the production templates..

Other Server Roles

.NET-based application servers will often interact with other server roles. These could include:

- SQL Server 2000
- Exchange 2000 Server
- Host Integration Server 2000
- BizTalk Server 2002
- Application Center 2000

You should be aware that each of these applications will have their own security considerations and you may need to modify the server role settings to allow them to run properly. In Chapter 12, "Customizing the Security Environment for Specific Applications," we show how to modify the settings to allow Application Center to support a specific application.

Note: *Security Operations for Exchange 2000 Server* demonstrates how to modify server roles to accommodate the requirements of Exchange 2000 Server.

Additional Security Measures

In addition to the enhanced security provided by the security templates, there are additional security measures that should be implemented on servers running .NET-connected applications. This section covers those measures.

IIS Lockdown

IIS servers provide a great deal of functionality. However, to make your IIS servers as secure as possible, you should restrict this functionality to only that which is required. The easiest way to do this is with the IISLockdown tool. IISLockdown is a highly configurable utility that allows you to specify the nature of your Web server. It will then remove any functionality that is not required for that type of Web server.

You should, of course, thoroughly test any changes before implementing them in a production environment.

Note: IISLockdown is available as part of the Security Toolkit and on the Microsoft Security Website. Further details can be found in the "More Information" section at the end of this chapter.

IISLockdown can perform many steps to help secure Web servers. These can include:

- Locking files
- Disabling services and components
- Installing URLScan
- Removing unneeded Internet Server Application Programming Interface (ISAPI) DLL script mappings
- Removing unneeded directories
- Changing ACLs

You can use IIS Lockdown to secure many types of IIS server role. For each server, you should pick the most restrictive role that meets the needs of your Web server.

IIS Lockdown Ini File

When you run the IIS Lockdown Wizard, you are presented with a list of product/role templates to choose from. A product/role template contains a default list of IIS security settings based on the product/role requirements as defined in the Iislockd.ini file. The settings for the Presentation Tier Server Role and Business Tier Server Role do not need to differ from one another; however there are differences between the production server roles and development server roles. This book includes, in Appendix D, the Iislockd.ini for two product/role templates.

- Production Server for .NET-based Applications – This product/role template removes the ability to debug Web applications/services, and locks down all other non-essential IIS features
- Development Server for .NET-based Applications – This product/role template leaves debugging of Web applications/services enabled, but locks down all other non-essential IIS features

► **To modify the Iislockd.ini file**

1. Extract **Iislockd.exe** to **C:\Iislockd** by typing the following command at a command prompt in the folder that contains Iislockd.exe:

```
iislockd /t:c:\iislockd /c
```

2. Open **C:\Iislockd\Iislockd.ini** in **Notepad**.

3. Locate the **ServerTypes=** line and at the end of the line add the following text:

```
, netDev, netProd
```

4. Copy the entire **[netDev]** section from either Appendix D or **C:\SecurityOps\Iislockd\iislockdini.txt** and paste it at the end of the Iislockd.ini file.
5. Copy the entire **[netProd]** section from either Appendix D or **C:\SecurityOps\Iislockd\iislockdini.txt** and paste it at the end of the Iislockd.ini file.
6. Save the **Iislockd.ini** file and close **Notepad**.

URLScan Template

URLScan is a tool that filters ISAPI extensions and is installed as part of the IIS Lockdown Wizard. URLScan screens all incoming requests to an IIS Web server, and only allows those that comply with a rule set defined in the Urlscan.ini file for the server role to pass. When URLScan is accurately configured, it is effective at reducing the exposure of an IIS server to possible attacks.

You should view the URLScan log file to determine the rules in effect for your Web server defined in the URLScan.ini file and find out which attacks have been blocked by URLScan. This file is located, by default, at \%windir%\system32\inetsrv\urlscan\urlscan.<*date*>.log. The following is an example log entry showing a blocked request for a file ending with .config.

```
[05-14-2002 - 22:19:12] Client at 127.0.0.1: URL contains extension '.config',
which is disallowed. Request will be rejected.  Site Instance='1', Raw URL='/
machine.config'
```

Note: URLScan 2.5 also provides the option to change the log file location using the **LoggingDirectory** attribute. By default, **PerDayLogging** is set to **1**, which creates a new log each day with the format urlscan.*mmddyy*.log.

Note: For changes to the URLScan.ini to take effect, the Web service must be restarted. Restarting the Web service overwrites the current URLScan log file. To preserve the log file, you should rename it before restarting the service.

Each product/role in the IIS Lockdown Wizard will have a corresponding URLScan ini file. This book includes the following URLScan ini files:

- Urlscan_netProd.ini – for the .NET Framework Production Server role
- Urlscan_netDev.ini – for the .NET Framework Development Server role

► **To add the Urlscan.ini files to the IIS Lock Down Wizard**

1. Extract the **SecOps.exe** file included with this book.
2. Copy **Urlscan*.ini** from **C:\SecurityOps\Iislockd** to **C:\Iislockd**.

Using the IIS Lock Down Wizard

The IIS Lock Down Wizard should be run on all .NET Framework Server roles. The .NET Framework Production Server role should be selected for all production Presentation and Business tier servers. The .NET Framework Development Server role should only be selected for servers being used for development.

After modifying the Iislockd.ini file and adding the URLScan ini files for the .NET Framework Server roles, you can then run the IIS Lock Down Wizard.

► **To secure servers with the IIS Lock Down Wizard**

1. Start **C:\Iislockd\IISLockd.exe**.
2. Click **Next**.
3. Select **I agree**, and then click **Next**.
4. Select the appropriate .NET Framework Server role, and then click **Next**.
5. Ensure **Install URLScan filter on the server** is selected and then **Next**.
6. Click **Next**.
7. If the **Digital Signature Not Found** dialog box appears, click **Yes**.
8. Click **Next**.
9. Click **Finish**.

ASP.NET Considerations

.NET-based applications will typically use ASP.NET as their runtime host. This section examines the specific measures you can take to secure applications running in an ASP.NET environment.

Note: More details on runtime hosts are contained in Chapter 1, "Introduction."

Configuration Files

Prior to ASP.NET, ASP configuration settings were generally stored in the IIS metabase. The metabase is a binary data store IIS uses to store and retrieve configuration settings. IIS uses its metabase much in the same way as Windows 2000 uses the registry. Relying on the IIS metabase has the following drawbacks:

- **Delayed updates**. Changes to the metabase generally required restarting IIS.
- **Difficult to replicate**. Not feasible to replicate configuration settings from one metabase to another.

While the ASP.NET configuration files replace nearly all the responsibilities of the IIS metabase for .NET-based applications using ASP.NET, there are still two functions that must be performed using the IIS metabase:

1. Creating web applications.
2. Custom mapping of file extensions not handled by ASP.NET.

To ease the configuration and deployment of ASP.NET applications, the configuration files for ASP.NET applications are stored in Extensible Markup Language (XML) files. Each file contains a nested hierarchy of XML tags and subtags with attributes that specify the required settings.

This means that you can add your own custom configuration settings using standard text editors to modify the configuration files. The system automatically detects changes to ASP.NET configuration files. You do not have to restart IIS or reboot the Web server for the settings to take effect.

When ASP.NET is installed, the default configuration settings are contained in a file called machine.config. This file contains configuration information about all the .NET-based applications running on a specific version of the .NET Framework. Settings in machine.config propagate downward to all virtual directories on the machine that use that particular version of the framework. It is located in the following path:

```
%windir%\ Microsoft.NET\Framework\{version}\config
```

Application specific configuration information is stored in a file called web.config. This file is stored in the same directory as the site content. At deployment time, you only need to copy the content directory to obtain both the content and the application configuration.

Note: Both the machine.config and web.config files must contain well-formed XML. XML is case sensitive, therefore it is very important to ensure the proper case is used for all entries in the .config files.

Configuration Inheritance

When a server receives a request for a particular Web resource, ASP.NET determines the configuration settings for that resource hierarchically, using all the configuration files located in the *virtual* directory path for the requested URL. For example, imagine a Web site with the following file structure, where the application root directory (approot) is also the application virtual directory.

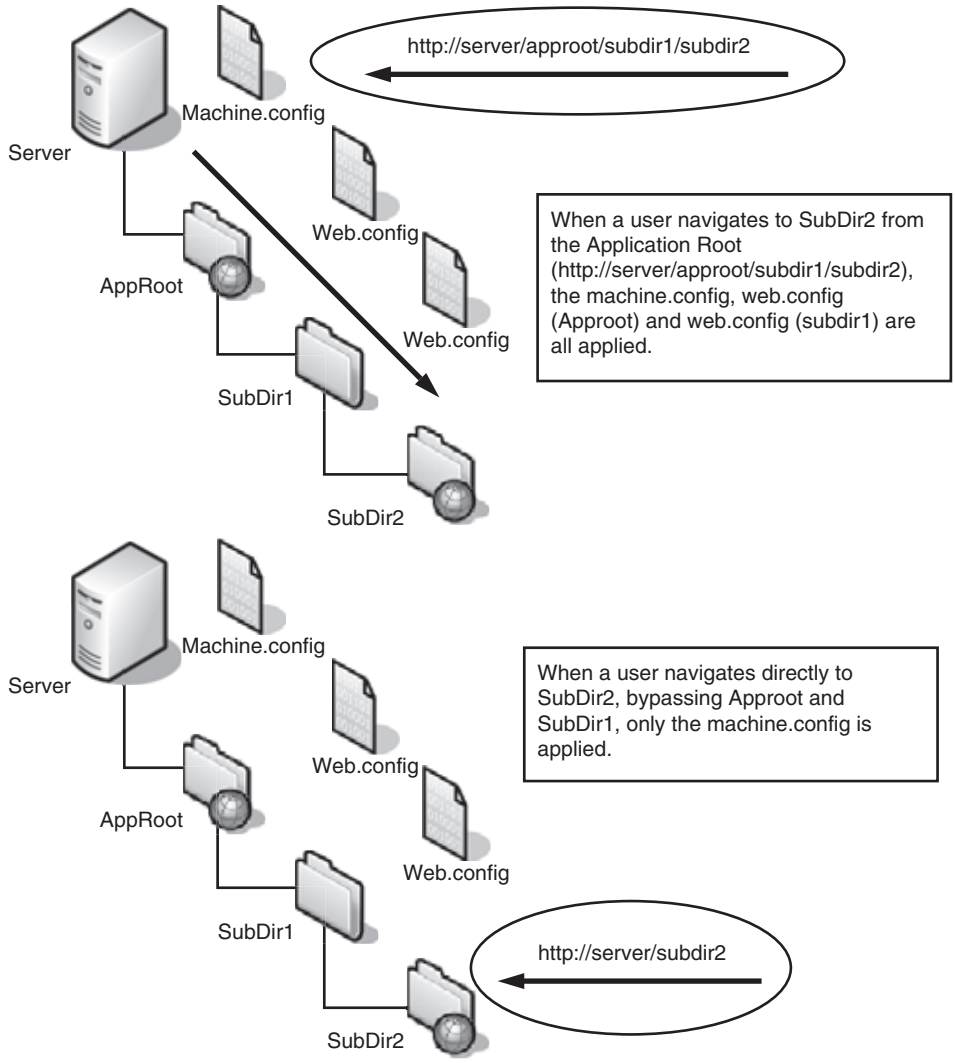


Figure 10.2
How ASP.NET determines configuration settings for a resource

Normally, configuration settings specified lower in the hierarchy will overwrite settings for the same section in parent directories. By selective use of web.config files, you can implement more granular security settings.

In the example shown above, you could use a web.config file in the SubDir1 directory, to restrict access to specific users, and SubDir2 will inherit those settings. Any user who accesses the resource using the path *http://server/approot/subdir1/subdir2* will inherit settings from both machine.config and the web.config file in SubDir1. However, configuration files are processed ONLY if they are contained in the URL path used to access them. This means that if SubDir2 is also a virtual directory and the

user access it directly (for example, *http://server/subdir2*) then none of the configuration information contained in the web.config file for SubDir1'will be applied. You should therefore be very wary of creating multiple virtual directories in the same physical path.

Note: The ASP.NET configuration system applies only to ASP.NET resources (those registered to be handled by ASP.NET using the Aspnet_isapi.dll). The configuration system does not provide authorization for non-ASP.NET resources by default. ASP, HTML, TXT, GIF, and JPEG files, for example, are accessible by all users. In the preceding example, if directory browsing is enabled and no other restrictions are in place, all users can view non-ASP.NET files located in the application root directory, SubDir1, and SubDir2.

ASP.NET Applications on Domain Controllers

.NET-based applications should not be run on domain controllers. For external applications the reasons for this are obvious—you should never expose your organization's domain to the Internet. Instead, separate your Web servers from the domain by forcing them to authenticate locally or by placing them in a separate forest. However, even internal-only .NET-based applications should not be run on domain controllers.

If you attempt to run an application that uses ASP.NET on a domain controller, it will not work. This is because, under IIS 5.0, ASP.NET runs its worker process (Aspnet_wp.exe) using a weak account (a local computer account, localcomputername\ASPNET) rather than a domain account or the SYSTEM account. Unless you modify the **<processModel>** section of the machine.config file to use a domain account or the SYSTEM account, your application will fail due to the unavailability of the worker process.

Note: For more information, see Knowledge Base article Q315158, "ASP.NET Does Not Work with ASPNET Account on Domain Controller."

Configuration File Settings

All configuration information resides between the **<configuration>** and **</configuration>** root XML tags. Configuration information between the tags is grouped into two main areas: the configuration section handler declaration area and the configuration section settings area.

Configuration section handler declarations appear at the top of the configuration file between **<configSections>** and **</configSections>** tags. Each declaration contained in a **<section>** element specifies the name of a section that provides a specific set of configuration data and the name of the .NET Framework class that processes configuration data in that section.

The configuration section settings area follows the **<configSections>** area and contains the actual configuration settings. There is one configuration section for each declaration in the **<configSections>** area. Each configuration section contains subtags with attributes that contain the settings for that section.

Note: We do not list all the possible settings in the config files here, just the ones that are important for security purposes. To find a more complete description of the config files, see the "ASP.NET Configuration" topic in the Microsoft .NET Framework SDK Documentation.

Protecting Sensitive Information in Configuration files

The ASP.NET configuration files require credentials to perform certain actions, for example to specify an account as an impersonation identity. By default these credentials are stored in plain text in the configuration files. However, by using the Aspnet_setreg.exe utility, you can encrypt and store these attribute values in the registry under a secure key. For more information see Knowledge Base article Q329290, "HOW TO: Use the ASP.NET Utility to Encrypt Credentials."

Machine.Config Settings

You should keep as many settings as possible in the machine.config file, rather than the web.config file. Consolidating configurations settings into a single file is more efficient than distributing them across multiple web.config files. However, there may be situations, where you need to use different web.config files for different Web applications. In circumstances where web.config files are necessary, maintain your baseline security settings in machine.config and overwrite, as necessary, in applications down the hierarchy. Because machine.config will contain the set of default security settings if settings do not exist at the application, machine.config must contain the highest, common denominator of security settings.

Administrators can enforce configuration policy decisions on a machine, preventing them from being overridden by application specific web.config files. This is done using the **<location>** configuration directive, which takes a path and supports an **allowOverride** attribute. The following XML fragment illustrates how to prevent the **<customErrors>** section from being overridden in the "Default Web Site".

```
<location path="Default Web Site" allowOverride="false">
  <system.web>
    <customErrors mode="On"/>
  </system.web>
</location>
```

Table 10.6: The path attribute can contain the following types of paths

Path Type	Comments
Web Site Descriptions	This is the site description, as found in the Description field on the Web Site type when viewing site properties in inetmgr. <no path> Omitting the path attribute altogether or using an empty string indicates that this applies to all sites on the machine. If an administrator uses this, they will need to remove the corresponding default configuration entry from its default location in machine.config.
URL Paths	Components used in application URL paths, for example, "/myapp" or "/myapp/mypage.aspx".

Unless you have good reason to do otherwise, you should set the **AllowOverride** to false. This prevents developers from adding new web.config files that are outside the security policy of your organization.

Note: The configuration lockdown mechanism applies only to configuration data. If an individual property can be set programmatically or has a corresponding page directive, the property can be overridden in that way. It is a useful guard against incorrectly configured applications but not against improperly written code.

Defining Specific Application Settings

Machine.config includes a pre-defined element **<appsettings>** which is used to provide information for specific applications. For example, if your .NET-based application needs to communicate with a SQL database, you would store the SQL logon information in here.

System.Web Section

<authentication>

The authentication element specifies how users are authenticated by the .NET-based application. Use the following syntax to define which method of authentication to use:

```
<authentication mode="Windows|Forms|Passport|None">
```

Which attribute you choose will depend upon your security policy and the requirements of the application.

If you use the Forms method of authentication, there is a **<forms>** subtag which you should use to specify the security used on the form data. The **<forms>** subtag takes a number of attributes, but one of the most important is **Protection**. This determines how the authentication cookie is protected as it is passed over the network. Wherever possible, you should set this attribute to **All**, which ensures that both validation

and encryption are used. Exactly which algorithms are used to protect the cookie is determined by the **<machinekey>** element, described later in this section.

The **<forms>** subtag itself contains another subtag—**<credentials>**. This is used to specify a username and password for authentication purposes. Wherever possible, you should avoid placing credentials in the XML file and use an external source, such as a database. However, if you must specify name and password credentials, you should also use the **passwordFormat** attribute to ensure that the password is encrypted using the SHA1 algorithm.

Note: Credentials can be stored in encrypted form in the registry. For more details, see Microsoft Knowledgebase articles Q329250, "FIX: Stronger Credentials for processModel, identity, and sessionState" and Q329290, "HOW TO: Use the ASP.NET Utility to Encrypt Credentials and Session State Connection Strings."

<compilation>

This element controls whether or not debugging is enabled in applications (debugging is disabled by default for greater performance), where temporary files generated are placed as the application is compiled, which compilers are available on the system, and which assemblies are used when an ASP.NET application is compiled.

It is very important that debugging is disabled on all production servers. Debugging information can contain source code of value to attackers, along with other information that increases the vulnerability of your servers. Fortunately, with a proper development and staging environment, there should never be an instance where debugging is required on production servers. Debugging and code alteration should be done on development servers, pushed to a staging server, tested there, and then finally pushed to production servers.

You may wish to change the directory where temporary files are generated while the application is compiled. This is done using the **tempDirectory** attribute. A default installation places all temporary files under a folder named "Temporary ASP.NET Files" under the installation root. The process identity used for ASP.NET must have full control over this folder.

Note: When debugging is enabled, ASP.NET generates files that include source code in the temporary files folder.

<customErrors>

Error messages can provide very useful information for attackers. An error message may, for example, display what type of authentication has been attempted, or even information about internal servers in an organization. For this reason, the **<customErrors>** element allows you to specify a simplified error message. A default message is provided, or you can create your own.

The **mode** attribute of this element determines when these simplified error messages will appear. By default, this attribute is set to **RemoteOnly**, which means that remote users will see the simplified error message, while local users will see the more detailed precise one. If the attribute is set to Off, then all users will see the detailed error message, if it is set to On, even local users will see the simplified message.

It is sometimes useful during development to set the mode attribute to Off which enables verbose error reporting for remote users. This allows a team to develop on shared servers. However, on production servers you should set this attribute to On, ensuring that the real error messages are not displayed even if there is a successful **IPspoof** attack. You should ensure that applications are deployed with On for the **<customErrors>** mode attribute. In fact, this setting is so important, that you should use the **<location>** configuration directive to ensure that it cannot be overridden by web.config files.

Note: Locking down individual sections of machine.config is covered earlier in this chapter.

<http Handlers>

This element is responsible for defining how files with particular file extensions are dealt with by ASP.NET. Whenever the .NET Framework is installed with IIS on a Windows 2000–or Windows XP–based machine, ASP.NET and its corresponding script maps are installed, as shown in the following table:

Table 10.7: Script Maps added when the .NET Framework is installed

Feature	Extensions	Description
ASP.NET Pages (web forms)	.aspx	ASP.NET pages
Web Services	.asmx	XML Web Services
.NET Remoting	.rem, .soap	Facilitates communication with objects in different application domains, whether local or remote. Applications can communicate over a choice of protocol channels including TCP sockets and HTTP using either binary or SOAP formatters.
ASP.NET Handlers	.ashx	Allows request processing logic to run that isn't associated with a page.
ASP.NET Tracing	.axd	Allows request/response capture and diagnostics. This generally is not a facility that should be exposed on a production server.
Protected Content Types	.asax, .ascx, ,asp .config, .cs, .csproj, .vb, .vbproj, .webinfo, .licx, .resx, .resources	These content types are registered to ASP.NET and are explicitly blocked from download by virtue of being mapped to the ASP.NET HttpForbiddenHandler.

If you have installed the .NET Framework, but do not intend to use ASP.NET functionality on a server, you should edit the Urlscan.ini file to block ASP.NET file extensions. If you run IIS Lockdown to disable ASP.NET, this will remove IIS script mappings that point to the ASP.NET ISAPI.dll, but this will result in ASP.NET files being displayed as text. This can be dangerous, as some ASP.NET files, such as .config files, may contain sensitive information such as usernames and passwords. As an additional precaution, if you have installed IIS Lockdown, but not removed the script mappings in IIS, you can map the ASP.NET extensions to the 404.dll to render them inoperable.

Note: The .asp content type is mapped to the ASP.NET **HttpForbiddenHandler** to guard against the situation where the .asp extension is mapped directly to the ASP.NET_isapi.dll without first updating and testing the .asp code to ensure that it will work correctly with ASP.NET.

<httpRuntime>

This element defines the environment that Web applications run in, including application timeout values, thread management settings, and the maximum length of the client request queue. You should closely examine all of these settings because appropriate configuration of many of them will help you to minimize the risk of denial-of-service attacks against the server. For example the **maxRequestLength** attribute determines the maximum size of a file that can be uploaded to the server. By default the setting is 4096 (4MB) but you may wish to decrease this value to avoid users uploading very large files.

Note: The maximum value of the **maxRequestLength** is 2GB.

It is worth noting that the setting defined by the **maxRequestLength** attribute can also be implemented using URLScan. URLScan defines a setting known as **MaxAllowedContentLength**. This specifies the maximum allowed numeric value in bytes of the content length request header, which would contain any uploaded files. The default setting is 2000000000, approximately 2GB.

Note: MaxAllowedContentLength does not always stop the server from reading in more data than is specified. For example, if a client makes a chunk transfer encoded POST, this URLScan option is not effective in tracking the size of the entity in the request.

<processModel>

This element determines the settings for the process in which ASP.NET applications will run. This element applies to all applications as it can only be configured within machine.config.

The most important attribute is **enable.** This determines whether ASP.NET-based applications run in the same process as IIS (enable = false), or in a separate process (enable = true). If a denial-of-service attack is launched against an ASP.NET-based application, and it is running in the same process as IIS, this could affect the stability of the entire Web server. You should therefore set the **enable** attribute to **true** wherever possible.

Note: If the **enable** attribute is set to false, all other settings in the **<processModel>** element are ignored and the settings for the IIS process are used instead.

You must restart to affect changes to the enable attribute.

Another useful attribute for this tag is **logLevel**. If you suspect a security breach that may involve an ASP.NET-based application, you should change the **logLevel** attribute from the default option of **Errors** to **All.** This will ensure that you receive maximum information about the events that occur in the ASP.NET process.

During an attack, you may need to reconfigure the **memoryLimit** attribute. This attribute specifies the maximum allowed memory size (as a percentage of total system memory) that the process can consume before ASP.NET launches a new process and reassigns existing requests. The default setting is 60 percent. So, if an attack is launched and results in abnormal memory usage by the process, the **memoryLimit** option limits the usage at 60 percent, or at the level set by the administrator.

Note: When ASP.NET is running under IIS version 6 in native mode, the IIS 6 process model is used and the settings in the **<processModel>** section are ignored.

<identity>

This element uses the **impersonate** attribute to define if impersonation is enabled. By default **impersonate** is set to false, so the trusted subsystem model is used. This means that access is checked at the gate and the request is executed using a single process identity (in this case the ASPNET account.) However, for some applications you may wish to set **impersonate** to true. This will enable the traditional Windows impersonation model, which ensures that the credentials of the user determine what resources the machine can and cannot access.

Note: For more information on identities, see Chapter 3, "Authentication and Authorization Design" in Building Secure ASP.NET Applications available from MSDN, see the "More Information" section for details.

If the user account is anonymous, then by default, the process will run under IUSR_*MachineName* where *MachineName* is the name of the Web server handling the request.

Note: If you set **impersonate** to true, you can also specify a particular username and password to use. Wherever possible you should avoid this so that you do not specify the username and password directly in the XML file.

<authorization>

This element allows you to specify users and groups of users that can access an application. The **<authorization>** element has two subtags, **<allow>** and **<deny>**. Use these tags to restrict your site as much as possible, only allowing the users, roles, and verbs that are absolutely necessary for site operation. For example, you can restrict GET actions for a particular directory to the administrator only, yet allow the Everyone group to POST. Users or Groups specified here should be of the form Domain\Username.

Note: You can also use wildcards for the **<allow>** and **<deny>** subtags. * refers to all users, while **?** refers to anonymous users.

<machineKey>

Data, such as cookie authentication data, generated by the server is encrypted and decrypted using settings contained in this element so that only the server can read the data. You can specify configuration settings for **machineKey** at the machine, Web site, and directory levels.

The **machineKey** element has three required attributes—**validationKey**, **decryptionKey** and **validation**. Both the **validationKey** and **decryptionKey** attributes are set to **AutoGenerate** by default. This generates a random key and stores it in the Local Security Authority (LSA). The **AutoGenerate** option is fine under many circumstances, but if you are implementing a Web farm on your Presentation tier or if you wish to isolate multiple applications on a single server, it is not appropriate. Under these circumstances all the servers are encrypting and decrypting the same values and so you must set the attribute manually using the value option.

Note: For more information on manually setting the value option see Knowledge Base articles Q313091, "HOW TO: Create Keys by Using Visual Basic .NET for Use in Forms Authentication" or Q312906, "HOW TO: Create Keys by Using Visual C# .NET for Use in Forms Authentication."

The validation tab determines which type of encryption is used for data validation, for example when cookies are passed from a form to authenticate a user. Wherever possible, this should be set to 3DES because this will provide the maximum level of encryption.

<pages>

This element controls some of the default behaviors for all ASP.NET pages. ASP.NET supports a feature called **ViewState** that allows server controls to persist their state across form posts using hidden HTML fields.

The default configuration for this setting is shown below:

```
<pages buffer="true" enableSessionState="true" enableViewState="true"
enableViewStateMac="true" autoEventWireup="true"/>
```

If it is necessary to maintain session state, wherever possible you should change **enableSessionState** to **ReadOnly**. Selecting **ReadOnly** ensures that state will be maintained. This protects against session stealing or spoofing. You should protect this setting on your servers by using the **<location>** directive described earlier in this chapter.

Note: Individual page settings can override the **<location>** directive, so pages should be scanned for references to **enableViewStateMac** as well.

<sessionState>

This element determines how a user's session will be maintained within an ASP.NET-based application. Options include storing the session in-process, out-of-process, and in persistent storage, such as SQL Server. If you specify state information to be stored in SQL Server, ensure that you do not provide SQL logon information in plain text. To avoid that situation in a Windows authentication environment, you should use a trusted connection. Otherwise you should avoid placing SQL Server logon information in plain text.

Note: For details on how to securely store SQL logon information, see Chapter 8, "ASP.NET Security" and Chapter 12, "Data Access Security" of Building Secure ASP.NET Applications. See also Microsoft Knowledgebase article Q329250, "FIX: Stronger Credentials for processModel, identity, and sessionState."

<trace>

ASP.NET supports a useful feature called tracing that provides information about requests and responses that have occurred on a server. During development and testing, tracing can be useful, but on a production server it may provide information to an attacker.

For production servers, administrators should ensure that the default setting hasn't been overwritten. The setting should be:

```
<trace enabled="false" />
```

You should also review your individual pages to ensure that tracing hasn't been overridden and enabled at the page level using the following syntax:

```
<!– look for this on production apps and remove it –>
<%@ Trace="true" %>.
```

On development servers, where trace information will typically be used, you should still prevent trace output from being displayed anywhere but the **localhost**. You can do this by setting the **localOnly** attribute to true.

<trust>

This element uses the **level** attribute to determine an acceptable security level for accessing the application code. In version 1.0 of the .NET Framework the only valid option is Full (no restrictions) so there is no need to change any settings here.

<securityPolicy>

This attribute defines valid mappings of named security levels to policy files. The security levels would then be used in the **<trust>** element listed above. However since the **level** attribute of the **<trust>** element only accepts the Full option, there is no need to change any settings here.

<webServices>

This element controls the settings of the XML Web services that use ASP.NET. Using the **<protocols>** subtag, you can determine which transmission protocols ASP.NET can use to interpret the data arriving from a client.

Table 10.8: Protocols supported by default

Protocol	Description
HttpGet	Enables the use of the GET method and query string variables to call web service methods. This is useful during the development phase for testing web services.
HttpPost	Enables the use of the POST method and form data to call web service methods. This is often useful during the development phase.
HttpSoap	Enables the use of SOAP messages using the HTTP POST method. This is the traditional way to call web services.
Documentation	Generates a friendly help page that describes the methods offered by the web service. The location of the code used to generate this help page is controlled by the **<wsdlHelpGenerator>** directive and this can be customized for a given application.

For many production servers using private Web services, you can disable all but the **HttpSoap** protocol. However, if your production servers are providing publicly discoverable Web services, you must also enable **HttpGet** and **HttpPost** to allow the service to be discovered. You can disable the Web service protocols by removing them from the **<protocols>** subtag in the **<webServices>** element, as is shown in the following code fragment:

```
<webServices>
    <protocols>
      <add name="HttpSoap"/>
```

```
            <!– disable all but HttpSoap
               <add name="HttpPost"/>
               <add name="HttpGet"/>
               <add name="Documentation"/>
            –>
        ...
        </webServices>
```

In the ASP.NET application model, Web pages intended for the browser use the .aspx extension. To differentiate XML Web services from regular ASP.NET pages, XML Web services use the .asmx extension. If an application is not using Web services, you can disable the .asmx extension altogether using the **<http Handlers>** element to map it to the **httpForbiddenHandler**.

Web.Config Settings

Wherever possible you should place your configuration settings in machine.config. However in certain circumstances you may need application specific configuration items in web.config. For example, you would use web.config to limit the access of specific pages in a particular application:

```
<!– Mark the AccountSummary.aspx page as available only to authorized users –>
    <location path="AccountSummary.aspx">
        <system.web>
            <authorization>
                <deny users="?"/>
            </authorization>
        </system.web>
    </location>
```

Removing Sample Applications

When you install the .NET Framework on production servers, you should ensure that you use the redistributable installation. This only contains the necessary runtime components and doesn't install sample applications or documentation. Sample applications are designed for learning and testing purposes only and are not designed to be secure in a production environment. If you do have any sample applications installed on your production servers, you should remove them. This applies to all sample applications, even those that can only be accessed from *http://localhost* or 127.0.0.1.

File and Folder Permissions

Although this is somewhat application specific, some general rules apply for setting ACLs on the various content types in application directories. It's often easiest to create separate directories for the various content types and use inherited ACLs on the directories that contain each type. You should modify the ACLs as shown for a .NET-based application.

Table 10.9: ACLs that should be modified on a server running a .NET-based application using ASP.NET

File Type	Access Control Lists
ASP.NET Pages or Services (.aspx, .asmx, .rem, .soap)	Everyone (X) Administrators (Full Control) System (Full Control)
ASP.NET User Controls (.ascx)	Everyone (X) Administrators (Full Control) System (Full Control)
Static Content (.txt, .gif, .jpg, .html)	Everyone (R) Administrators (Full Control) System (Full Control)

ASP.NET uses a special subdirectory under an application called the bin directory for business logic. The ASP.NET process identity and the set of identities under which the application runs need read and execute permissions to this directory. This directory typically contains .dll files: it should never be granted read, script, or browser access in the IIS metabase.

In some cases, if you have enabled impersonation for your applications, they will need to access one of the managed compilers in the .NET Framework while impersonating a (typically) anonymous user. If this is the case for your applications, you will need to alter the ACLs on csc.exe (the C# compiler) or vbc.exe (the Visual Basic compiler) to allow execution for these identities by the IUSR account or equivalent. In addition, cvtres.exe is used by compilers to perform functions such as converting .RES (resource) files into a binary format. You may therefore also need to modify the ACL on this file.

The three files are located at:

```
%windir%\ Microsoft.NET\Framework\{version}\vbc.exe
%windir%\ Microsoft.NET\Framework\{version}\csc.exe
%windir%\ Microsoft.NET\Framework\{version}\cvtres.exe
```

Summary

In this chapter, we have examined the security of the servers running .NET-based applications. As part of your overall security strategy it is important that these servers are as secure as possible. Before deploying your applications, you should test them in a locked-down environment, using as a starting point the Development Server security templates we have provided. Once the application is ready to deploy, you should ensure that the servers which will run the application in production are also locked down, using (and modifying where necessary) the Production Server security templates provided.

More Information

For the complete *Security Operations for Microsoft Windows 2000 Server*:

http://www.microsoft.com/technet/security/prodtech/windows/windows2000/staysecure/DEFAULT.asp

For the *Security Operations for Exchange 2000 Server*:

http://www.microsoft.com/technet/security/prodtech/mailexch/opsguide/default.asp

Details on enabling success auditing for logon events filling the security log:

http://support.microsoft.com/default.aspx?scid=kb;en-us;Q316685

To obtain the IIS Lockdown tool:

http://www.microsoft.com/technet/security/tools/tools/locktool.asp

Details on troubleshooting and configuring IIS Lockdown and URLScan:

http://support.microsoft.com/default.aspx?scid=kb;en-us;Q309677

Details on the ASP.NET Configuration files

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconformatofconfigurationfiles.asp

Building Secure ASP.NET Applications

http://msdn.microsoft.com/library/en-us/dnnetsec/html/secnetlpMSDN.asp?frame=true

Windows 2000 Service Packs

http://www.microsoft.com/windows2000/downloads/servicepacks/default.asp

Knowledge Base article Q315158 ASP.NET Does Not Work with ASPNET Account on Domain Controller

http://support.microsoft.com/default.aspx?scid=kb;en-us;Q315158

Knowledge Base article Q329290: HOW TO: Use the ASP.NET Utility to Encrypt Credentials

http://support.microsoft.com/default.aspx?scid=kb;en-us;Q329290

Knowledge Base article Q313091 HOW TO: Create Keys by Using Visual Basic .NET for Use in Forms Authentication

http://support.microsoft.com/default.aspx?scid=kb;en-us;Q313091

Knowledge Base article Q312906 HOW TO: Create Keys by Using Visual C# .NET for Use in Forms Authentication

http://support.microsoft.com/default.aspx?scid=kb;en-us;Q312906

Knowledge Base article Q329250: FIX: Stronger Credentials for processModel, identity, and sessionState

http://support.microsoft.com/default.aspx?scid=kb;en-us;Q329250

Knowledge Base article Q323640: HOW TO: Secure the IUSER_<Computer_name> Account

http://support.microsoft.com/default.aspx?scid=kb;en-us;Q323640

Knowledge Base article Q300432: HOWTO: Promote a Member Server Running IIS to a DC Running IIS

http://support.microsoft.com/default.aspx?scid=kb;en-us;Q300432

11

Network Considerations for Servers Running .NET-Based Applications

In the previous chapters, we examined how to secure the servers involved in running .NET-connected applications. However, if you are to create as secure an environment as possible, you also need to examine the communications between the different computers in your environment.

In this chapter, we show a structured process that allows you to define the elements in your network that communicate, and decide how best to protect the data flow in your organization. We also examine different methods of authentication and how to implement them securely.

Defining the Network

The first step in providing effective network security is to detail the network you must protect. Defining the network involves identifying the following items:

- Physical Components
- Logical Components
- Traffic flow between the servers

By performing this analysis, you can determine which network traffic to protect, and identify which security methods to use.

Identifying Physical Components

Typically, a multi-tier architecture model is used to segment the physical computers required to support an application. The components of an application are spread across these tiers, and do not necessarily map directly to them; instead they correspond to the logical design of the application. For the purposes of this book, we define a three-tier architecture model, which supports the needs of most applications. The three-tier model is defined in Chapter 1, "Introduction."

Using Server Roles

Within each tier of the architecture, a number of computers are generally deployed to support the functionality of the application. These will have a number of different roles. Identifying the server roles involved in supporting the functionality of your .NET-based application is important, as communication between those servers will differ according to their functionality. More information on server roles can be found in Chapters 9 and 10 of this book and in *Security Operations for Windows 2000 Server*.

Once you have determined which computers are required to deploy your .NET-based application, you need to establish the number of computers involved in the deployment and their physical location. The network devices that data must cross as it passes between computers will determine what protocols may be used to protect against data inspection and modification.

A .NET-based application, exposed to the Internet, is most often comprised of three physical segments: the external network (typically the Internet), the perimeter network (also known DMZ or demilitarized zone), and the private network, as shown in the diagram:

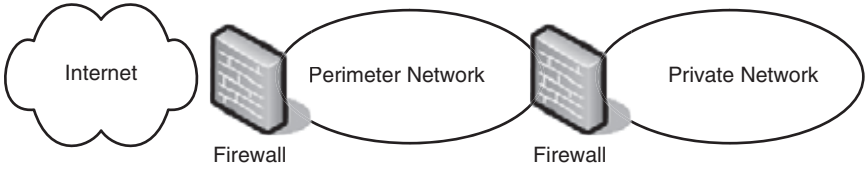


Figure 11.1
Physical segments supporting a typical .NET-based application

Typically, when deploying a three-tier application, the Presentation tier exists in the perimeter network, and the Business and Data tiers reside on the private network.

Identifying Logical Components

In addition to the server roles already described, a .NET-based application typically consists of a number of logical components. These communicate with one another to provide the functionality of the application. Most frequently these components are located where they are used. For example, if a component displays information to the client, it generally goes on the user's computer. If a component updates information in a database, it resides on the database server in the Data tier.

In many cases, there will be flexibility in where the logical components can be placed. If this is the case, you should consider whether the security of a component can be enhanced by placing it on a server rather than a client, or in the Data tier on the private network rather than in the Presentation tier in the perimeter network. However, you should not just consider the security of the component itself. In some cases you may choose to place a component in the Presentation tier, because placing

it in the Data tier would require you to loosen security significantly at the firewall to allow communication.

As you identify where the logical components of the application are placed on the network, you will be able to establish the network traffic that you need to secure. You should:

- Identify all network segments containing logical components of the .NET-based application.
- Identify where availability requirements designate that the servers need to be deployed in a load-balancing or clustered configuration.
- Identify all firewalls on the network. For each firewall, determine implemented protocol and packet filters.
- Identify if a firewall implements Network Address Translation (NAT) between network segments. NAT will limit the encryption and integrity protocols that may be used to protect transmitted data.
- Determine the server placement based on information stored on the server. For example, if the SQL database contains important data, it should be stored on the private network, not in the perimeter network.

Identify Traffic Flow

Once you determine the placement of the physical and logical components required for the .NET-based application, you need to identify the data flow between those components. The actual flow of data will show where to implement IP-layer and application-layer security to prevent data interception and modification.

The type of data protection to implement will depend on your organization's written security policy and the type of data being transmitted.

When identifying traffic flow, look for the following:

- *How data passes between computers.* Include how data is routed between the servers as an order or task is processed.
- *Which server(s) host each logical component.* If the components are clustered between servers, note that the data can be sent to any cluster node.
- *Protocols used to transmit data between network layers.* Some protocols, such as SMTP and HTTP, can be protected with application layer security by implementing SSL. RPC communication may be encrypted on an application by application basis. Other protocols, such as Telnet, cannot take advantage of SSL and must be protected with Internet Protocol Security (IPSec).
- *Does the data pass through NAT devices?* If NAT is performed, you cannot use IPSec to protect the traffic, as IPSec protection prevents NAT translation of IP and TCP/UDP headers.

- *Does the data pass through firewalls*? The written security policy will determine if a firewall can inspect content with an intrusion detection system or if data must be transmitted through the firewall in an encrypted state.

Secure Traffic Flow

Once you determine traffic flow between the servers supporting your .NET-based application, you need to decide what security should be applied to transmitted data. Your options include preventing data modification, ensuring that data is transmitted from a trusted source, and protecting the data from inspection.

Protocols to Secure Transmitted Data

Several network transmission methods can be used to protect transmitted data. The protocols that can be used include:

- Server Message Block (SMB) Signing
- Secure Socket Layers (SSL)
- Internet Protocol Security (IPSec) using Authentication Headers (AH)
- IPSec using Encapsulating Security Payloads (ESP)

We will examine each of these in more detail.

Server Message Block (SMB Signing)

SMB signing ensures the data sender's authenticity by enforcing mutual authentication of the client and the server. When the data is transmitted, a digital security signature is placed into each SMB and verified by both the client and the server. This provides protection against data modification and man-in-the-middle attacks.

SMB signing is supported by computers running Windows 98, Windows Me, Windows NT 4.0, Windows XP and Windows 2000 and can therefore be used if you only have Windows clients. Within each operating system, you can choose to enable or disable SMB signing. If SMB signing is enabled at the Windows 2000-based server, only clients that enable SMB signing will be able to connect.

Table 11.1: Procedures to enable SMB signing,

Operating System	Procedure
Windows 98 and Windows Me (Client only)	Follow the procedure outlined in Q230545: How to Enable SMB Signing in Windows 98
Windows NT 4.0 (Client or Server)	Follow the procedure outlined in Q161372: How to Enable SMB Signing in Windows NT

Operating System	Procedure
Windows 2000 Professional and Windows XP Clients	Enable the following local policy or Group Policy settings: \Computer Configuration\Windows Settings\ Security Settings\Local Policies\Security Options: Digitally sign client communications (always)
Windows 2000 Servers	Enable the following local policy or Group Policy settings: \Computer Configuration\Windows Settings\ Security Settings\Local Policies\Security Options: Digitally sign server communications (always)

Secure Socket Layers (SSL)

SSL provides encryption of data transmitted between the client and the server. The data is encrypted using a symmetric key generated at the client-side of the transmission. The symmetric key is in turn encrypted with the public key of the server's asymmetric key pair. This means that the server is the only entity that can decrypt the symmetric key, using its private key.

Note: Mutual authentication is possible when the server enforces authentication of the client. The server is authenticated when the certificate of the server is validated.

SSL encryption protects data at the application layer. An SSL-enabled application will encrypt the data before it is sent to the TCP/UDP layer of the IP protocol stack. SSL encryption may only be used by the application if it is coded to implement SSL encryption. For example, a Web server can implement SSL security, while a Telnet server cannot.

When SSL encryption is enabled at a server, the server will often listen for connections on a port other than the default port for the unencrypted protocol. The table shows the standard and SSL-enabled ports for common applications that support SSL encryption:

Table 11.2: Standard and SSL Ports for common protocols

Protocol	Standard Port	SSL Port
Post Office Protocol v3 (POP3)	TCP 110	TCP 995
Internet Message Access Protocol v4 (IMAP4)	TCP 143	TCP 993
Hyper Text Transfer Protocol (HTTP)	TCP 80	TCP 443
Lightweight Directory Access Protocol (LDAP)	TCP 389	TCP 636
Simple Mail Transfer Protocol (SMTP)	TCP 25	TCP 25
Network News Transfer Protocol (NNTP)	TCP 119	TCP 563

Obtaining Web Server Certificates

To enable SSL on a Web site, you must first acquire a Web Server Certificate from a certificate authority (CA) that chains to a CA in your computer's trusted root store. The Web Server Certificate can be obtained from a private CA; a commercial CA, such as VeriSign or RSA; or hosted by your company. The decision as to where to obtain the Web Server Certificate will depend on the exposure of the application, as well as the type of transactions.

If the application is being accessed from the Internet, you will normally obtain a certificate from a commercial CA for your Presentation tier servers. Most organizations and individuals trust certificates issued by a commercial CA, whereas trust of certificates issued by another company's private public key infrastructure (PKI) must be manually trusted.

On the other hand, if the Web server hosts a .NET-based application that will only be accessed by internal users, you can deploy a Web Server Certificate from a private CA. Within Active Directory, you can publish the trusted root certificate so it is trusted by all forest members.

If your Web Server Certificate must contain custom extensions, such as application policy or certificate policy extensions, you will normally need to issue the certificate from a private CA structure. Commercial CAs do not generally deploy custom certificates without great expense.

Note: When requesting a Web Server Certificate from either a private or commercial CA, ensure that the certificate request includes the Web server's fully qualified domain name (FQDN). The FQDN must be the DNS name used by Web clients to connect to the Web server. The subject name of the Web Server Certificate should also match the DNS name requested by the Web client. If it does not, a warning appears.

Installing an SSL Certificate for a Web Site

After you obtain the SSL certificate, you need to install the certificate on the Web server.

► **To enable SSL encryption with a certificate issued by a private enterprise CA**

1. From **Administrative Tools**, open the **Internet Services Manager** console.
2. In the console tree, expand ***Computer*** (where *Computer* is the NetBIOS name of the Web server computer), right-click **Default Web Site**, then click **Properties**.
3. In the **Default Web Site Properties** dialog box, in the **Directory Security** tab, click **Server Certificate**.
4. On the **Welcome to the Web Server Certificate Wizard** page, click **Next**.
5. On the **Server Certificate** page, click **Create a new certificate**, then click **Next**.
6. On the **Delayed or Immediate Request** page, click **Send the request immediately to an online certification authority**, then click **Next**.

7. Provide name and key details for the Web Server Certificate request by performing the following steps:
   a. On the **Name and Security Settings** page, enter a friendly name for the certificate, key length, and whether to enable Server Gated Cryptography (SGC). Click **Next**.
   b. In the **Organization Information** page, enter naming information about the organization and the OU, then click **Next**.
   c. In the **Your Site's Common Name** page, enter the Web site's FQDN, then click **Next**.
   d. In the **Geographical Information** page, enter country/region, state/province and city/locality information, then click **Next**.
8. In the **Choose a Certification Authority** page, choose the online enterprise CA to which you wish to submit the certificate request, then click **Next**.
9. In the **Certificate Request Submission** page, review the certificate request parameters, then click **Next**.

Note: The chosen CA will either issue or deny the certificate request based on the issuance requirements and permissions of the Web Server Certificate template.

10. In the **Completing the Web Server Certificate Wizard**, click **Finish**.

The procedure is different if the request is submitted to a commercial CA. In this case, the certificate request must be generated and then submitted to the commercial CA for validation.

Note: For maximum security in an SSL environment, you should use 128-bit SSL encryption for all connections.

► **To enable SSL encryption with a certificate issued by a commercial CA**

1. From **Administrative Tools**, open the **Internet Services Manager** console.
2. In the console tree, expand ***Computer*** (where *Computer* is the NetBIOS name of the Web server computer), right-click **Default Web Site**, then click **Properties**.
3. In the **Default Web Site Properties** dialog box, in the **Directory Security** tab, click **Server Certificate**.
4. On the **Welcome to the Web Server Certificate Wizard** page, click **Next**.
5. On the **Server Certificate** page, click **Create a new certificate**, then click **Next**.
6. On the **Delayed or Immediate Request** page, click **Prepare the request now, but send it later**, then click **Next**.

7. Provide name and key details for the Web Server Certificate request by performing the following steps:
   a. On the **Name and Security Settings** page, enter a friendly name for the certificate, key length, and whether to enable SGC, then click **Next**.
   b. In the **Organization Information** page, enter naming information about the organization and the OU, then click **Next**.
   c. In the **Your Site's Common Name** page, enter the Web site's FQDN, then click **Next**.
   d. In the **Geographical Information** page, enter country/region, state/province and city/locality information, then click **Next**.
8. In the **Certificate Request File Name** page, provide a name for the certificate request file, then click **Next**.
9. In the **Request File Summary** page, review the certificate request parameters, then click **Next**.
10. In the **Completing the Web Server Certificate Wizard**, click **Finish**.
11. Send the certificate request file to the commercial CA.

The commercial CA will process the certificate request, performing any required background and validity requests to ensure the certificate request is from the correct source. If the commercial CA issues your Web server a Web Server Certificate, you must install it by performing the following steps:

1. In the **Internet Services Manager** console, in the **Default Web Site Properties** dialog box on the **Directory Security** tab, click **Server Certificate**.
2. On the **Welcome to the Web Server Certificate Wizard** page, click **Next**.
3. On the **Pending Certificate Request** page, click **Process the pending request and install the certificate**, then click **Next**.
4. In the **Process a Pending Request** page, designate the certificate response file provided by the commercial CA, then click **Next**.
5. In the **Certificate Summary** page, review the details of the provided Web Server Certificate, then click **Next**.
6. In the **Completing the Web Server Certificate Wizard**, click **Finish**.

Enabling SSL Encryption

After installing the SSL certificate on your Web servers, you need to enable SSL encryption for the Web site. This is accomplished by modifying the properties of the Web site itself.

Note: You should always enable SSL encryption at the Web site level, rather than the page level. Typically, SSL-encrypted Web pages require authentication and may allow weaker forms of authentication such as basic authentication. Only by enabling SSL encryption for the entire Web site can you ensure that weak credentials are not encrypted with SSL.

► **To enable SSL encryption for a Web site**

1. From **Administrative Tools**, open the **Internet Services Manager** console.
2. In the console tree, expand ***Computer*** (where *Computer* is the NetBIOS name of the Web server computer), right-click **Default Web Site**, then click **Properties**.
3. In the **Default Web Site Properties** dialog box, on the **Directory Security** tab, ensure that the **View Certificate** button is enabled.
4. On the **Directory Security** tab, in the **Secure Communications** box, click **Edit**.
5. In the **Secure Communications** dialog box, click **Require secure channel (SSL)** to enable SSL encryption for the Web site.
6. If the Web site hosts sensitive or confidential information, enforce 128-bit encryption by clicking **Require 128-bit encryption,** then click **OK**.
7. Click **OK** to accept Inheritance Overrides.

Note: For more information on how SSL encrypts data transmitted between the SSL-protected server and the client computer, see Q245152: "How Secure Sockets Layer Works."

Internet Protocol Security (IPSec)

Internet Protocol Security (IPSec) allows the implementation of security at the network layer, rather than at the application layer, as implemented by SSL.

IPsec provides two choices of security service:

- **Authentication Header (AH)** – this provides authentication of the computers involved in transmitting data, protection against modification of data, and preventing unauthorized computers from participating in the network.
- **Encapsulating Security Payload (ESP)**–this provides both authentication of the computers involved in a network data transmission and encryption of the transmitted data.

IPSec can use both integrity and encryption algorithms to protect the transmitted data. The following table shows the available integrity and encryption algorithms for IPSec in a Windows 2000 environment:

Table 11.3: IPSec Integrity and Encryption Algorithms

IPSec Protocol	Integrity Algorithms	Encryption Algorithms
AH	MD5 SHA1	Does not use
ESP	MD5 SHA1	NULL DES 3DES

Each algorithm differs in strength and therefore the amount of security it provides:

- **Secure Hash Algorithm version 1 (SHA1)** is a cryptographic message digest algorithm that takes a message of less than 2^{64} bits in length and produces a 160-bit message digest.
- **Message Digest version 5 (MD5)** is a cryptographic message digest algorithm that takes a message of arbitrary length and produces a 128-bit message digest.
- **NULL** indicates that no encryption is applied to the data. Typically, null encryption is only used for interoperability testing to conform to the IPSec RFCs.
- **Data Encryption Standard (DES)** is an encryption algorithm that encrypts data with a 56-bit randomly generated symmetric key.
- **Triple DES (3DES)** is a variation on the DES encryption algorithm where DES encryption is applied three times to the plaintext. The plaintext gets encrypted with key A, then with key B, and finally with key C. The most common form of 3DES uses only two keys. The plaintext gets encrypted with key A, then with key B, and finally with key A again.

To ensure that the strongest integrity and encryption algorithms are implemented, you should implement AH with SHA1 integrity and ESP with both SHA1 integrity and 3DES encryption.

Internet Protocol Security Using Authentication Headers (AH)

IPSec using AH provides authentication of the two endpoints in a data stream and protects against data modification as it moves across the network. AH can only be used to allow connections from trusted hosts. If a host is unable to negotiate an IPSec security association using AH, the host cannot connect to the server.

IPSec using AH adds a digital signature to the data packets. The signature is calculated against all non-mutable fields in a packet, and is then stored in the header. This will prevent IPSec AH packets from crossing network devices that implement NAT, as NAT translates the IP header and TCP/UDP header for all outbound and inbound packets. This translation invalidates the calculated digital signature.

Note: All IPSec with AH communications will use transport mode. Transport mode provides end-to-end encryption between two computers implementing IPSec. The data is encrypted or signed at the sending computer, then decrypted or verified by the recipient computer.

IPSec Using Encapsulating Security Payloads (ESP)

IPSec using ESP not only provides authentication of endpoints, but also encryption of transmitting data. ESP encrypts the data payload so that only the endpoint in the security association is able to decrypt the information. This protects the data against inspection attacks.

Unlike SSL encryption, IPSec encryption can be implemented for any application, as the encryption is applied at the IP layer of the TCP/IP protocol stack. This means that the application is not aware of the encryption and decryption process as data transmits. However, as with IPSec with AH, IPSec with ESP requires that a NAT device is not present between the client and the server.

Note: If an application can support SSL encryption, you should generally implement SSL encryption rather than IPSec encryption.

IPSec using ESP encrypts and signs the original IP packet payload. Three fields are added to the original IP packet:

- **ESP header**. The ESP header contains the Security Parameter Index used to encrypt data and the sequence numbers used to reassemble the data at the recipient end.
- **ESP Trailer**. Includes padding so the length of the application data and the ESP trailer is evenly divisible by 32 bytes.
- **ESP Authentication Trailer**. This is an integrity check value that contains the digital signature used to prevent modification and validate the sender.

IPSec using ESP can apply both an encryption algorithm and an integrity algorithm to the transmitted data. For the strongest protection, consider using 3DES as the encryption algorithm and SHA1 as the integrity algorithm.

Note: The integrity algorithm employed by IPSec using ESP does not sign the entire packet. To apply the digital signature to the entire packet, you must apply both ESP and AH protection to the IPSec-protected data.

Implementing IPSec

For servers that are not part of Active Directory, you will need to apply IPSec policies directly. However, for those which are part of Active Directory, you can use Group Policy to apply IPSec policies. This will help ensure consistent application of the IPSec policies. In an OU structure based on server roles, you can apply different IPSec policies to specific server roles.

For the OU structure demonstrated in Chapter 10 of this book, you would apply IPSec Policies as shown in the table.

Table 11.4: IPSec Policies

Policy	Where applied	Settings
Client	Domain or Clients OU	Client (Respond Only)
Server	Production Servers OU	Custom (with AH, ESP or ESP and AH) respond (with AH, ESP, or ESP and AH) if requested)

To restrict which computers can connect to the .NET application servers, you can place the approved client computers in a specific OU, and only apply the client (Respond Only) IPSec policy to that OU.

Note: The encryption calculations for IPSec are processor-intensive. To decrease the amount of processor time spent encrypting and decrypting data, consider implementing IPSec-offload cards on your servers. These cards move all cryptographic calculations to the network card from the CPU of the server.

To implement IPSec, an authentication method must be applied to computers in the IPSec security association. Windows 2000 and Windows XP support three methods for IPSec security association authentication:

- **Shared secret**. A text string is configured at each endpoint in the IPSec security association. If the same text string is implemented at each endpoint, the computers are considered authenticated.
- **Kerberos**. Computers in the same Active Directory forest can authenticate using the default Kerberos authentication protocol. The computer accounts use Kerberos to validate the identity of the other computers in the security association.
- **Certificate-based**. Computers that have certificates that chain to the same trusted root CA can authenticate with each other by presenting certificates as proof of identity.

In a pure Windows 2000 single forest environment, you would normally choose Kerberos as the means of authentication. However, in many cases, not all the computers will be members of the same Active Directory forest, or some will be members of a Windows NT 4.0 domain. Under these circumstances you will need to use certificate-based authentication to authenticate the two computers in the IPSec security association. You should never use shared secret authentication for IPSec security associations unless interoperability requirements, such as a Unix host, do not support Kerberos or certificate-based authentication. If the shared secret is compromised at one client computer, all connections to the .NET application server are compromised.

Note: For more information on how to implement IPSec to protect transmitted data, see the "Step-by-Step Guide to Internet Protocol Security (IPSec)" and "Using IPSec to Lock Down a Server", both listed in the More Information section at the end of this chapter.

Choosing Between IPSec and SSL

Both IPSec and SSL are effective at encrypting data as it passes across the network. So, given the choice, which do you pick? Consider the following to help you choose:

- If the transmitted data must pass through a NAT device, then implement SSL encryption, if available. IPSec cannot pass through NAT devices because the methods used to protect the transmitted data are broken by NAT.
- IPSec is only supported by Windows 2000 and Windows XP clients. If you must implement data transmission security for Windows 9x or Windows NT clients, you must implement SSL encryption.
- If an application supports SSL encryption, it is easier to configure and implement SSL encryption rather than IPSec encryption.
- If a server hosts several applications that implement SSL, it may be easier to implement a single IPSec solution, rather than configure each application to implement SSL encryption.
- IPSec provides more flexibility in configuration than SSL. With SSL, you can only enforce a 128-bit encryption key length. In IPSec, you can choose whether to implement AH, ESP, or a combination of both and choose specific integrity and encryption algorithms.

Protecting Against Attack

The protocols described here will help you to guard against attacks on network traffic. Exactly which you use will depend on the specific nature of the environment you are protecting, the security policy of your organization, and the likely threats you face.

Data Integrity Attacks

Your organization's security policy may require that transmitting data be protected against modification. This prevents data modification during transit over the network, but does not prevent data inspection. There are two methods for preventing data modification: SMB signing and IPSec with AH. In addition to preventing modification, these methods provide mutual authentication of the client and the server for data transmission.

As data travels from one part of your organization to another, you need to be sure that it has not been altered. To protect against data modification or to prevent unauthorized connections to a server, you can implement IPSec AH. IPSec with AH is only of use in an environment when you know that both the clients and the servers are Windows 2000 or Windows XP. In a mixed environment, where Windows 98, Windows Me or Windows NT 4.0 clients must access the servers, you would implement SMB signing to prevent data modification.

Data Inspection Attacks

Even though you may be relatively sure that data is passing through your organization unaltered, some data will be so sensitive that even the wrong person reading the data could be considered a major breach of security.

If the network traffic uses a protocol that supports SSL encryption, you should implement SSL rather than IPSec encryption. If it does not, you should implement IPSec with ESP encryption. However, remember that under these circumstances the data will not be able to pass through a NAT device.

If your organization's security policy requires that the entire contents of a data packet are protected against both modification and inspection, you should implement IPSec with both AH and ESP. The default ESP digital signature does not include the original IP header in the signed portion of the packet.

Note: For information on other types of possible attacks, see Chapter 2, "Understanding Security Risk" of *Security Operations for Windows 2000 Server*.

Increase Security with ISA Server

Implementing a Microsoft Internet Security and Acceleration (ISA) Server between the Internet and your Presentation tier servers can increase the security of your .NET-based applications by not exposing the servers directly to the Internet. In addition, ISA can implement application layer filters that inspect the content of an HTML stream to ensure that only approved verbs and content requests are submitted to.NET application servers.

Web Publishing with ISA

ISA Server allows you to secure a Web server that implements SSL by using Web Publishing to publish all HTTPS requests to the HTTP server. Web Publishing allows two configurations: You can choose to publish the HTTPS traffic as HTTP or you can establish a second SSL session and publish the HTTPS traffic as SSL-protected traffic.

Note: You cannot use host headers when using SSL because HTTP requests that use SSL are encrypted. Host headers are part of the encrypted request and cannot be interpreted and routed to the correct site.

The configuration you choose will depend on the security policy of your organization. For example, if your organization's security policy requires end-to-end encryption of sensitive data, you must publish the SSL traffic using SSL between the ISA server and the Web server.

As an example, consider a Web site that uses a Web cluster of three computers with Network Load Balancing. The following configuration for the Presentation tier

servers is required to use Web Publishing to make the Web site available on the Internet.

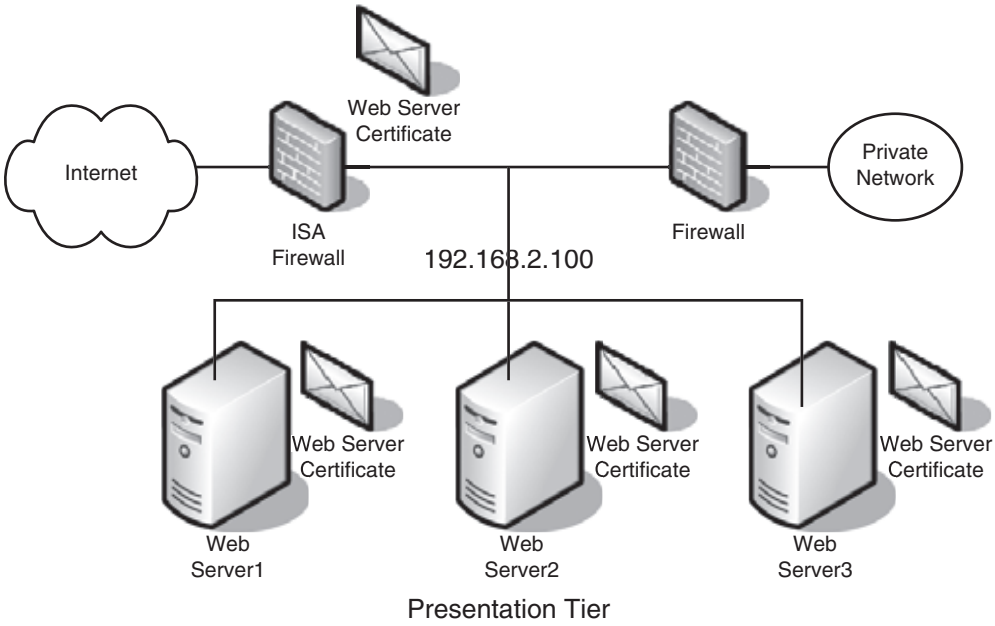


Figure 11.2
Using Web Publishing to make Presentation tier servers available on the Internet

You must obtain the certificate at the ISA firewall from a commercial CA. The subject of the certificate must match the DNS name used to access the application from the Internet.

The certificate on the Presentation tier Web cluster nodes can be obtained from a private CA hosted on the private network. The subject of the certificate must match the DNS name used by the ISA firewall when it connects to the Presentation tier Web cluster IP address of 192.168.2.100. You should implement the same certificate at all three nodes.

You should ensure that the Certificate Revocation List (CRL) and Authority Information Access (AIA) from the internal CA hierarchy are published to locations accessible from your perimeter network. The ISA server must access these URLs to validate certificates at the Presentation tier Web cluster nodes.

Note: For more information on how certificate status checking is performed, see the white paper, "Troubleshooting Certificate Status and Revocation," available at *http://www.microsoft.com/technet/security/prodtech/dbsql/tshtcrl.asp.*

In this configuration, you should configure a Web Publishing rule at the ISA server stating that all traffic destined to the external interface of the ISA server is published to the Presentation tier Web cluster IP address (192.168.2.100). In this scenario, the incoming SSL traffic is bridged as SSL requests.

This configuration ensures that end-to-end data encryption is provided. It implements a second SSL session between the ISA server and the selected node of the Presentation tier Web cluster. For the configuration to work properly, you must ensure that your .NET-based application maintains state information in case the Web cluster fails over to a different node in the Web cluster, otherwise all session data will be lost.

If the organization's security policy does not require end-to-end encryption or that host headers must be implemented to determine what site to display for incoming traffic, the SSL requests can be redirected as HTTP requests. In this scenario, the Web server certificate and private key must only be installed at the ISA server. This ensures that SSL encryption is enforced between the Web client and the ISA server over the Internet, but HTTP traffic is passed from the ISA server to the Presentation tier Web server cluster.

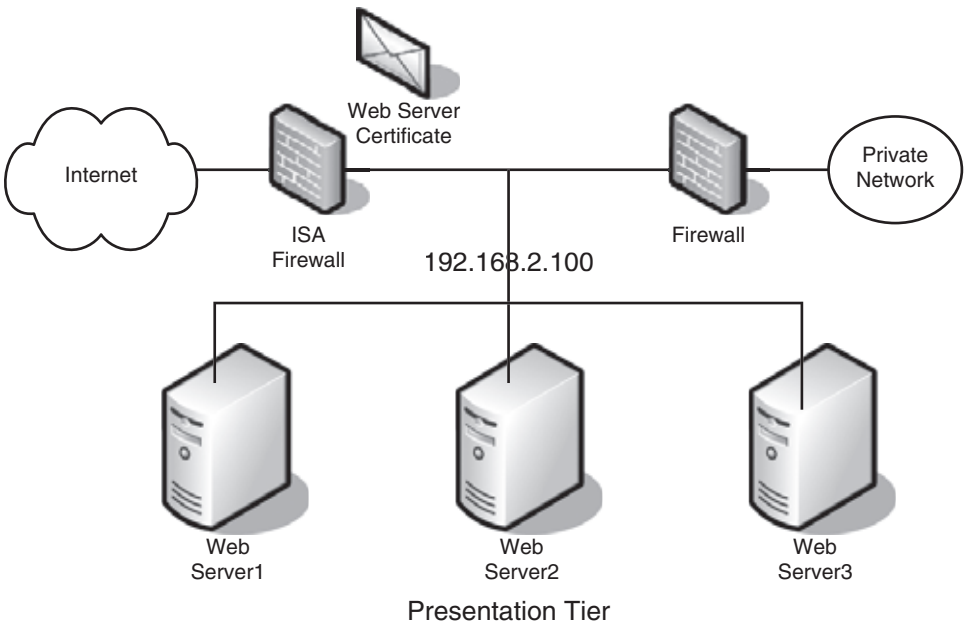


Figure 11.3
Web Publishing where SSL requests are redirected as HTTP Requests

In this configuration, the certificate at the cluster nodes must be obtained from a commercial CA. The subject of the certificate must match the DNS name used to access the application from the Internet. This DNS name must map to the ISA firewall's external IP address, not the Presentation tier Web cluster IP address.

Note: Both methods allow application filters to inspect content at the ISA server before data is transmitted to Presentation tier servers. For more information on using application filters with ISA, see, "Implementing Application Filters" later in this chapter.

Server Publishing with ISA

A second method that can be used to publish Presentation tier Web servers protected by an ISA server is to implement Server Publishing. Server Publishing maps TCP port 443 on the external interface of the ISA server to TCP port 443 of the internal Presentation tier servers. The benefit of this method is that the Web server certificate does not have to be imported at the ISA server. The disadvantage is that all traffic is routed from the Internet to the Web cluster in the Presentation tier without content inspection.

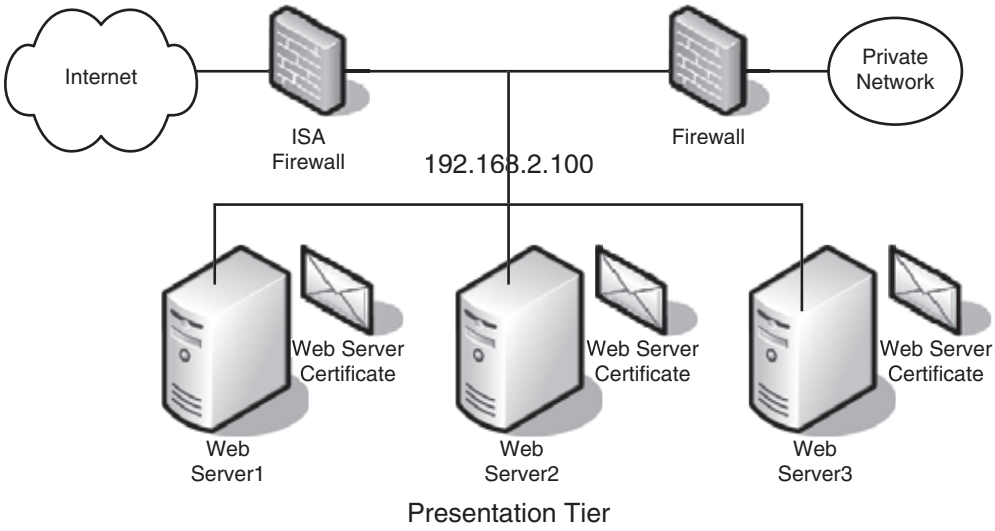


Figure 11.4
Using Server Publishing to make Presentation tier servers available on the Internet

In this configuration, the Server Publishing rule is configured to publish all inbound HTTPS requests to the external IP address of the ISA firewall to the Web cluster common address of 192.168.2.100. Because the inbound HTTPS requests are simply routed to the Web cluster, the Web Server Certificate only has to be installed at the Web cluster nodes, not at the ISA server.

The Web Server Certificate implemented at the ISA server must be obtained from a commercial CA. The subject of the certificates must match the DNS name used to access the .NET-based application from the Internet. This DNS name must map to the ISA firewall's external IP address, not the Web cluster's IP address. Server Publishing routes the HTTPS traffic as HTTPS to the Web cluster's IP address.

Server Publishing is configured in two parts at the ISA server. First, a custom protocol definition for inbound HTTPS traffic must be defined. In this example, the protocol definition is named Web Secure Inbound.

Then, a Server Publishing rule must be established that publishes all Web Secure Inbound traffic from the external IP address of the ISA server (131.107.154.14) to the IP address of the Web cluster in the Presentation tier (192.168.2.100).

Note: To enable ISA Web Publishing or Server Publishing to a Web site, you must remove IIS from the ISA server so that the ISA server is not listening for connections on TCP port 80 and TCP port 443.

Implementing Application Filters

If you implement Web Publishing to make Web sites available through an ISA server, you may want to implement application filters at the ISA server to perform application content inspection.

The application filters are implemented at the ISA server as ISAPI filters. Two filters that may be considered for environments including .NET-based applications include:

- **URLScan ISAPI filter**. This will detect and block undesired HTTP verbs and application extensions at the ISA server. You will need to ensure that the URLScan.ini file is configured to allow the required verbs and application extensions for your .NET-based application. URLScan and URLScan.ini are covered in more detail in Chapter 10.
- **BizTalk XML Filter**. This custom ISA application filter examines XML documents from an HTTP POST request and verifies the XML data before routing the documents to back-end systems. By verifying the XML data at the ISA server, this filter provides an additional layer of protection for the computers in the organization running Microsoft BizTalk® Server.

Note: Full documentation of the ISA Server XML Web Filter sample code can be found at www.microsoft.com/isaserver/development/2000/XMLWebFilters.asp.

Guidelines for Increasing Security with ISA Server

The following table outlines the decision matrix for choosing between Server Publishing and Web Publishing to protect Web servers in your perimeter network.

Table 11.5: ISA Publishing Rule Guidelines

If your security policy requires	Implement the following Publishing type
End-to-end encryption with no content inspection by ISA	Server Publishing
End-to-end encryption with content inspection by ISA	Web Publishing, with HTTPS traffic redirected as HTTPS traffic
Encryption from client to network perimeter with content inspection by ISA	Web Publishing, with HTTPS traffic redirected as HTTP traffic

Once you decide on the publishing configuration, you must determine where to obtain the Web Server Certificates for SSL encryption and where to install the certificates to protect the .NET-based application.

- If you are implementing Server Publishing, the Web Server Certificates must be obtained from a commercial CA so the .NET-based application is trusted by external customers. The Web Server Certificate must be installed at each node in the Presentation tier Web cluster. The subject name of the certificate must resolve to the ISA server's external IP address.

  Note: The commercial CA may require the purchase of three separate certificates, one for each node in the Web cluster.

- If you are implementing Web Publishing, where HTTPS traffic is redirected as HTTPS traffic:
  - Obtain a Web Server Certificate for the ISA server from a commercial CA. The subject name of the certificate must resolve to the ISA server's external IP address
  - Obtain Web Server Certificates for each node of the Presentation tier Web cluster from a private CA hosted on the private network. The same Web Server Certificate may be installed on each node.
  - The ISA server must be able to resolve the CRL and AIA extension URLs in the certificates issued to the Web cluster. The ISA server uses the URLs to validate the certificates issued to the Web cluster nodes.
- If you are implementing Web Publishing, where HTTPS traffic is redirected as HTTP traffic, obtain the Web Server Certificate for the ISA server from a commercial CA. The subject name of the certificate must resolve to the ISA server's external IP address.

As well as configuring Web and Server Publishing appropriately on your ISA server you should also consider other measures you can take to improve the security and performance of the ISA server. You should disable file caching at the ISA server between the Internet and the perimeter network. This will prevent attacks against the ISA server to view contents stored in the file cache of the ISA server.

To help with performance if you are implementing Web Publishing at the ISA server, you should consider implementing an SSL Accelerator network card at the external interface of the ISA server. If you are performing SSL Bridging, so that HTTPS requests are redirected as HTTPS requests, you should consider implementing SSL Accelerator cards at each node in the NLBS cluster.

Note: Further guidelines and security measures for the perimeter ISA server are prescribed in the "Microsoft Systems Architecture: Internet Data Center" guide, available at *http://www.microsoft.com/technet/itsolutions/idc/default.asp*.

Protecting Authentication Credentials

To ensure the security of your environment, it is very important to know that users are who they claim to be. In some cases, .NET-based applications will only be accessed anonymously, but in the case where users are authenticated, you will need to make sure that they do so as securely as possible. This verification process is also applied to applications running in the security context of a user. Typically, authentication is implemented when the user provides an account and password combination, but there are a number of ways it may be achieved.

Note: Authentication is not the same as authorization. Authentication is the process of identifying the user that is accessing a resource. Authorization is the process of determining whether that user or computer has the necessary permissions to access the resource.

Note: Once authentication has occurred, there is still data which may be present on the network and may reveal information about a user. For example, a cookie may be used to present a user's credentials to other parts of the same Web site. For more information on this topic, see *Building Secure ASP.NET Applications*.

Available Authentication Methods

The following methods of client authentication may be used by .NET-based applications.

- Anonymous Authentication
- Basic Authentication
- Digest Authentication
- Integrated Windows Authentication
- Certificate Authentication
- Microsoft .NET Passport

We will look at each of these in turn.

Note: The authentication methods for the application should be determined when the .NET-based application is designed and developed. For developer information on secure authentication, see *Building Secure ASP.NET Applications*.

Anonymous Authentication

Anonymous authentication gives users access to the public areas of your Web site without prompting them for a user name or password. When a user attempts to connect to your public Web site, your Web server assigns the user to the Windows user account called IUSR_*computername*, (where *computername* is the name of the server on which IIS is running). There is no need for this account to be a member of the Domain Users group, and you should ensure that it is not.

If you have multiple sites on your server, or if you have areas of your site that require different access privileges, you can create multiple anonymous accounts—one for each Web or FTP site, directory or file. By giving these accounts different access permissions, or by assigning these accounts to different Windows user groups, you can grant users anonymous access to various areas of your public Web and FTP content.

Basic Authentication

Basic authentication allows a user to provide a user account and password combination. The password is Base64-encoded before being sent over the network. Basic authentication is supported by most Web browsers, but the password is highly susceptible to interception due to the weak protection of the password during transmission.

If you implement Basic authentication, ensure that you implement SSL encryption. This ensures that if the password is intercepted it cannot be decoded. Protect the entire Web site with SSL encryption, not just specific pages, as once provided, Basic authentication credentials are attached to every page request for a Web site, including those not protected by SSL.

Digest Authentication

Digest authentication protects a user's credentials by taking the user's password and other information about the user's request to the Web server and creating a *hash*, which is sent to the Web server. The hash is compared to a version of the hash stored in the user's properties in Active Directory; if the two hashes match, the user is authenticated. Digest authentication must be supported by the browser and can pass through NAT devices.

Digest authentication requires that the user's password is stored in reversible encryption at the domain controller. This weakens the password security of the domain controller and requires elevated physical security of the domain controller.

If reversible encryption is enabled for a user account, the user must change his or her password after the account option is implemented to store a hash of the user's password in Active Directory.

Integrated Windows Authentication

Integrated Windows authentication can use both Kerberos v5 authentication and Windows NT Challenge Response authentication. The current user's credentials are sent to the Web server. If the Web server cannot identify the user, the user must provide a user account, password, and domain information. The credential information is not sent across the network; instead, a hash of the authentication credentials is transmitted.

Integrated Windows authentication provides strong security of credentials, but cannot pass through firewalls. It is best suited for an intranet environment, in which both user and Web server computers are in the same domain.

Certificate Authentication

Certificate authentication allows a user to present a digital certificate as a form of authentication. The Web server map verifies that the certificate is *mapped* to an account either in the Web server's Security Accounts Manager (SAM) database or in Active Directory. The presented certificate is validated to ensure that it can be used for client authentication, is trusted, is not revoked, and is time valid.

Note: For more information on how certificate status checking is performed, see the whitepaper, "Troubleshooting Certificate Status and Revocation," available at *http://www.microsoft.com/technet/security/prodtech/dbsql/tshtcrl.asp.*

When you implement certificate authentication, you can implement the certificate mapping in either IIS or in Active Directory. Typically, you should choose to do the mapping in IIS if the IIS server is a not a member of an Active Directory domain or the mapping is only required at that single IIS server. Implement the certificate mapping in Active Directory if the certificate mapping is required at two or more IIS servers or by other applications that implement certificate mapping.

Note: You can enforce certificate authentication by configuring IIS to require certificates for user authentication and disabling all other forms of authentication in the **Authentication Methods** properties of the Web site.

Microsoft .NET Passport

Microsoft .NET Passport is a user-authentication service that lets users create a single sign-in name and password for easy, secure access to all .NET Passport-enabled Web sites and services. .NET Passport-enabled sites rely on a .NET Passport

central server to authenticate users rather than hosting and maintaining proprietary authentication systems. All .NET Passport sign-in and core profile cookies are strongly encrypted. Each participating Web site receives a unique encryption key that ensures information privacy.

Passport authentication removes an organization's responsibility to maintain a user account database from the organization. The accounts are maintained by Microsoft.

Custom Authentication

Custom authentication uses an application database, such as a SQL database or a Microsoft Commerce Server directory, to authenticate the user account. The method by which user credentials are transmitted depends on the database or directory storing the account information.

You should ensure that any custom authentication methods do not transmit user credentials in plaintext on the network. By using the SMS Network Monitor, you can observe if the credentials are passed in plaintext over the network wire.

If a custom authentication server does not support SSL encryption, consider using IPSec to encrypt the authentication data. IPSec cannot be used if the perimeter firewall implements NAT. The IPSec protection must terminate at the external interface of the firewall in the case of NAT implementation.

Summary

As you examine the security of your .NET-based applications, it is not enough to simply look at the servers on which the applications run. Generally these applications will run in a network environment and will generate network traffic that you also need to secure. You should look at all aspects of your internal and external networks and decide how best to secure network traffic flow to maintain the security of the data generated by your applications, along with taking measures to prevent applications being attacked over the network. If you follow the guidelines indicated in this chapter, you should go a long way to reaching that goal.

More Information

Security Operations for Windows 2000 Server

http://www.microsoft.com/technet/security/prodtech/windows/windows2000/staysecure/default.asp

Building Secure ASP.NET Applications

http://msdn.microsoft.com/library/en-us/dnnetsec/html/secnetlpMSDN.asp?frame=true

Troubleshooting Certificate Status and Revocation
http://www.microsoft.com/technet/prodtechnol/winxppro/support/tshtcrl.asp
Q245152: How Secure Sockets Layer Works
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q245152
Step-by-Step Guide to Internet Protocol Security (IPSec)
http://www.microsoft.com/windows2000/techinfo/planning/security/ipsecsteps.asp
Using IPSec to Lock Down a Server
http://www.microsoft.com/technet/itsolutions/network/maintain/security/ipsecld.asp
Q230545: How to Enable SMB Signing in Windows 98
http://support.microsoft.com/default.aspx?scid=kb;en-us;230545
Q161372: How to Enable SMB Signing in Windows NT
http://support.microsoft.com/default.aspx?scid=kb;en-us;161372

12

Customizing the Security Environment for Specific Applications

The previous 4 chapters have discussed settings that will apply to the majority of applications built on the .NET Framework. This chapter shows how to customize the settings according to the needs of a specific application. As an example, we use the FMStocks application first discussed in Chapter 1, "Introduction," showing the steps performed to ensure that the application can be operated as securely as possible.

Note: This chapter is supplemental to the recommendations made in the previous chapters. You should read those chapters prior to this chapter.

Determining Specific Application Requirements

As part of your security policy, for applications that are developed in house, you should require your developers to document in detail the way in which the application will function. This will include the following:

- The Services the application installs and requires
- How the application interacts with the File System and the Registry
- Any Group Policy security options required by the application
- Network protocols used by the application

If you are not fortunate enough to have all this information available directly from the developers, you will need to find it out in order to customize the security templates so that your application will function properly and be secure.

Using a Test Environment to Determine Requirements

An effective test environment is very important to ensure you can securely develop applications. However, it is equally important to make sure that you have a good test environment to help you work out how to securely deploy applications. This test environment should mirror as closely as possible the actual environment in which you will be deploying the application. At the very least it should include all the elements of the application and all the server roles that the application will communicate with. You can then use this test environment to determine how to customize your environment to support specific applications.

The test environment should be locked down as shown in Chapters 9 and 10 of this book. You can then attempt to install and run the application (and anything else required by the application), gradually loosening the security of the environment to support the functionality required by the application. When you have finally made all the changes required, these changes can be used to create customized security templates for the application.

The flowchart shows the steps you should follow to modify the security for a particular application in a test environment.

In the rest of this section, we will describe these steps in more detail.

Note: If you are installing multiple applications, you should repeat this process for each application. Once completing the process for the first application, you should lockdown the test environment using the customized templates for the first application and then repeat the rest of the process.

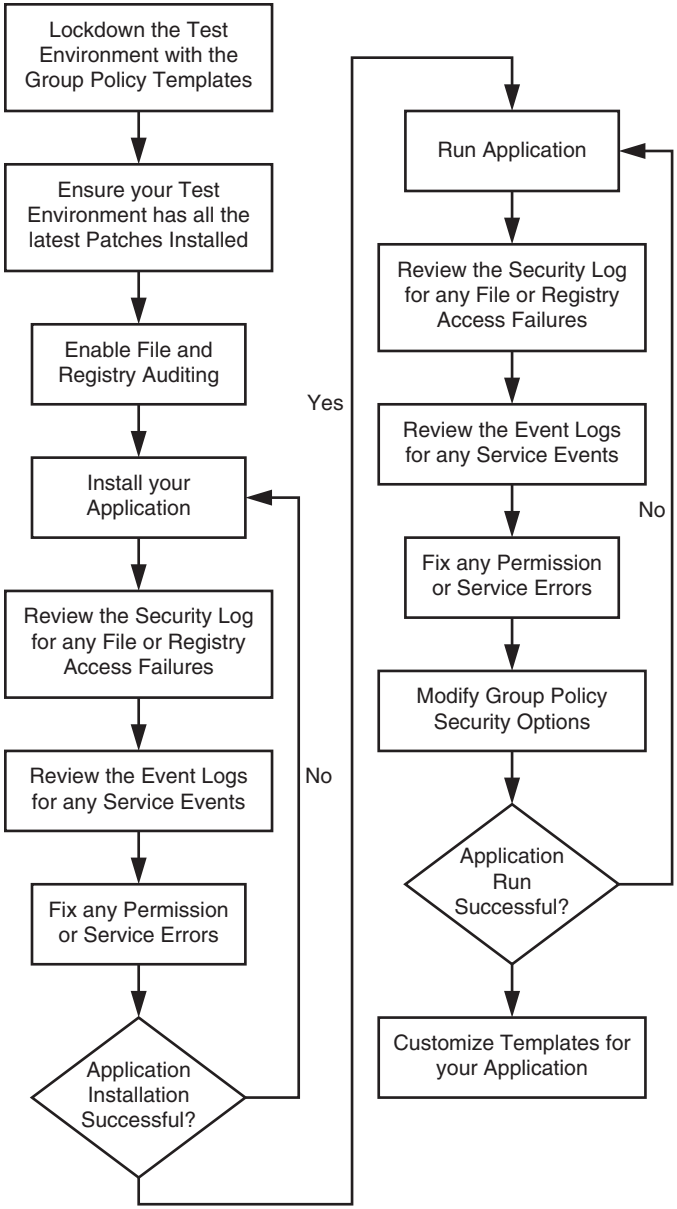


Figure 12.1
Steps to increase the security of an application

Patch Management

Patches are installed in an environment on a case-by-case basis. If there is no need to install a particular patch, because it doesn't affect your environment, then you should not do so. This means that if you are modifying your environment to get a particular application to work, you may need a patch that was previously not required. As part of your patch management process, you should be checking to see if any patches are required for your modified environment. If one of your applications fails to install or run after you have apparently made all the changes necessary, you may be missing a patch essential to its operation.

Do not fall into the trap of assuming certain patches are unnecessary because, for example a server is inside a firewall. With many vulnerabilities being exploited by worms, you should consider any server with even an indirect path to the Internet as Internet facing.

Auditing for File and Registry Access

As part of the Member Server Baseline Policy, auditing is significantly increased from the default settings that come with Windows 2000. However, although both success and failure events are audited for Object Access, we do not specify which file and registry objects to audit, so by default, none will be tracked.

Before installing and running your application in your test environment, you should enable failure auditing for all types of access on both the file system and the registry. This will allow you to track any permissions problems for your application.

You can define the specific files and registry entries to audit using security templates. We recommend using the security template, **enableaudit.inf**, included with this book as a starting point and then applying it using secedit. **Enableaudit.inf** enables auditing for the file system and certain registry keys. The following objects will be audited for failure of all types of access.

- System Drive and all subdirectories
- HKLM\System\Current Control Set\Services and all subkeys
- HKLM\Software and all subkeys
- HKLM\Users and all subkeys

Note: Depending on the configuration of your servers, you may wish to alter **enableaudit.inf** (for example, if you have multiple logical drives).

Depending on your environment, you may choose to apply these changes using Group Policy, or on the individual servers using the secedit command. Information on how to apply security templates using Group Policy can be found in Chapter 9, "Securing Servers Based on Role", and information on applying security templates using secedit can be found in Chapter 10, "Securing .NET Framework Servers Based on Role."

Configuring Auditing Using Security Templates

The specific file and registry objects to audit are defined by modifying the security descriptors within the template accordingly. Security descriptors are represented in the template by a language called SDDL, or Security Descriptor Definition Language. The following is a break down of a security descriptor record in a security template.

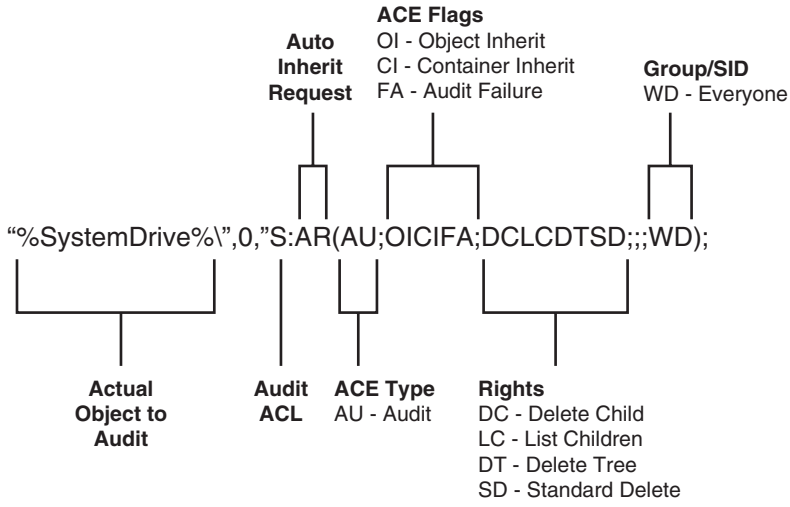


Figure 12.2
Annotated example of a Security Descriptor Record

The diagram shows the syntax of SDDL for an audit request that would configure audit failures for any file writes or deletes for the entire system drive (typically the C drive). Inheritance is propagated to all files and subfolders below the root, so every file and folder will receive this audit entry. Only writes and deletes will be monitored using the well known group Everyone. By using the Everyone group, any write or delete failures from any user will be logged in the Security Event Log. If you get too many records in the event log then you may want to change the Everyone group represented by "WD" in Figure 12.2 to the actual security identifier (SID) of the user you need to track and monitor.

Note: Inheritance is only changed to the Security Access Control List (SACL) while the Discretionary Access Control List (DACL) is left untouched.

Note: For more information on SDDL see the book *Writing Secure Code by Michael Howard and David LeBlanc (ISBN: 0735615888*) and from the Platform SDK, "Security Descriptor String Format" on MSDN (see the More Information section for further details)

Installing the Application

You are now in a position to install all the elements of your application in your test environment. It is entirely possible that an install may fail at this point. You should carefully monitor the install to see that it proceeds as it should. Even if the installation appears to succeed, you should carefully check the event logs for failure audits. These could include a failure to access the file system or registry, or to run, or install particular services.

If the application fails to install properly, you will need to change the security settings on the file system, the registry and/or services, and try the installation again. As you make these changes, be sure to note them carefully, as they will later be used to modify your security templates.

Note: If your application installs device drivers, you should ensure that any device drivers are properly signed, as the Member Server Baseline Policy prevents the installation of unsigned device drivers.

Running the Application

It is one thing to be able to install an application, quite another to be able to successfully run it. You should thoroughly test all the functionality of the application. Many times the developers or testers will be able to help you here, as they may have test scripts used in the final stages of application development.

As you test the functionality of the application, continue to monitor carefully the event logs for errors. These should help you troubleshoot any problems you encounter. Typically the problems will be with the settings defined for File and Registry Permissions, Services that are disabled, or the Group Policy security options. You should make the appropriate changes to the environment until the application runs properly, again carefully noting the alterations you have made.

You must remember to ensure that your application will continue to function properly for each category of user that would interact with it.. In particular, different users may have different requirements for an application. For example. you should make sure that that user accounts responsible for backing up the application can do so, those responsible for administering the application can do that, as well as the general users of the application.

Modifying the Security Templates

By now you should have determined the settings that allow the application to run successfully in your environment. You now need to modify the security templates to allow these settings to be applied.

To modify the security templates from the default templates supplied with this book, use the following procedure:

► **To modify the security templates from the defaults**

1. In the **C:\SecurityOps\Templates** folder, make a copy of **NETAppBusiness.inf** or **NETAppPresentation.inf** and name it for your application, such as **Application1Business.inf**.
2. Open **Application1Business.inf** and **Baseline.inf** in **Notepad**.
3. Copy the lines from **Baseline.inf** that you determined require modification for your application and paste them into **Application1Business.inf**.

   Note: If you are copying any lines from **Baseline.inf** that are from sections that are not in **Application1.inf** be sure to also copy the section heading, in brackets.

4. Save and close the **Application1Business.inf**.
5. Start the **MMC** and add the **Security Templates** add-in.
6. Right click **Security Templates** and select **New Template Search Path**.
7. Navigate to **C:\SecurityOps\Templates** folder and then click **OK**.
8. Expand **C:\SecurityOps\Templates** and select **Application1Business**.
9. Select the sections where you added entries to the file for customization, double click each entry, and modify as appropriate.
10. Repeat step 9 for each entry that needs to be modified.
11. When all required changes are completed, right click **Application1Business**, and then select **Save**.
12. Close the **MMC**.

Applying the Templates and Final Testing

Now that you have created the new security templates, you should rebuild your test environment and apply the new templates. If you test the install and the functionality of the application once again, you should find that everything now works as it should. Because you have changed the security of the environment in order to get the application to work, you should also thoroughly test the security of your new environment.

Disabling File and Registry Auditing

If your test environment has multiple purposes, you may need to disable the file and registry auditing that you previously enabled. For this purpose we have included the file **disableaudit.inf**.

Specific Network Protocol Requirements

Once you have succeeded in getting the application to function correctly, you can examine how to secure the communications generated by the application. You should use the guidelines given in Chapter 11 as the basis for increasing the security for the specific application. Make sure that you identify all communication traffic that takes place between the servers supporting the .NET-based application, so that you can identify where network protocol security is required, and what methods may be used to protect those traffic streams.

Customizing Security for FMStocks

FMStocks is a sample application which we are using to illustrate many of the concepts of this book. It is a multi-tier .NET-based application which requires changes to the default security environment specified in the previous chapters in order for it to function properly. In this section we show how we have customized the security environment to allow the FMStocks application to work.

Note: Before reading this section, ensure that you have read Chapter 1, "Introduction," as it contains important details on how FMStocks functions.

Application Center

The FMStocks infrastructure uses Microsoft Application Center 2000 to provide Network Load Balancing. When you install Application Center, it will make several changes to your environment. It will, for example, create and modify a number of accounts.

Table 12.1: Accounts created and modified by Application Center 2000

Group/account name	Description
Group: **ACA**_*machinename*	Application Center group. This local computer account is used for logging and other administrative operations.
User: **ACC**_*controllername*	Member of **ACA**_*machinename*. This cluster controller account is created on each cluster member. This account is used to manage cluster communication — it authenticates across servers, replicates content, and administers the cluster servers.
User: **ACL**_*machinename*	Member of **ACA**_*machinename*. This local utility account is used by an Application Center server. The server does not necessarily need to be in a cluster (that is, a server that will be used for staging). It is used for administrative work related to the server, such as writing event log information to Application Center Event and Performance Logging.
User: **IUSR**_*machinename*	IIS uses this Windows 2000 local account to authenticate anonymous users on the Web site. This account is not used in an Application Center cluster.

You should expect to see the creation of these accounts audited in the security log. In addition, the **ACL_computer** account is assigned to the "Log on as a batch job" user right as part of the installation process. This change should also appear in the Security log.

Note: In the Application Center MMC snap-in, if you receive an access denied error when attempting to open any property pages you will need to reinstall the WMI Pseudo component by running wmipsudo.exe. This is found in the **\Program Files\Microsoft Application Center** folder.

FMStocks Security Templates

The security templates included in Chapter 10, "Securing .NET Framework Servers Based on Role" need to be modified to allow the FMStocks application to run properly. In the case of this particular application, the changes necessary are identical for both Presentation and Business server roles.

Auditing

FMStocks uses Application Center for Network Load Balancing. Application Center generates a large amount of logon activity, and with the default settings in the Member Server Baseline Policy, this generates a very large number of success events in the Security log, which can result in the log filling too quickly and shutting down the system.

To get around this problem, we configure the following settings in the FMStocks incremental policies:

Table 12.2: FMStocks incremental audit policy settings

Policy	Computer Setting
Audit account logon events	Failure
Audit logon events	Failure

Services

The infrastructure supporting FMStocks includes both Application Center and Microsoft Operations Manager (MOM). There are a number of services that are required to allow these applications to run. The services configured are shown in the table on the next page.

Table 12.3: Services configured by the FMStocks incremental policies

Service	Startup Type	Reason for inclusion in FMStocks Presentation tier incremental
Distributed Transaction Coordinator	Automatic	Required by Application Center
Windows Management Instrumentation	Automatic	Required by Application Center
Application Center Cluster Service	Automatic	Configures the Application Center server array
Application Center Name Resolution Service	Automatic	Application Center uses to resolve computer names to IP addresses
Application Center Synchronization Service	Automatic	Application Center uses to synchronize content across the server cluster
Application Center Administration Service	Manual	Provides administrative support for Application Center
Application Center Log Query Helper	Manual	Used for querying Application Center performance and event data
OnePoint	Automatic	MOM agent used for monitoring the servers
NT LM Security Support Provider	Automatic	Required by the MOM agent
ASP.NET State Service	Disabled	FMStocks does not use out-of-process session states
IPSEC Policy Agent	Automatic	Needed to implement IPSec policy on server

Folder Permissions

Two of the user accounts created by the Application Center installation process require permissions to the Application Center folder. By default, the Application Center folder is located at %SystemDrive%\Program Files\Microsoft Application Center.

Table 12.4: Permissions required for Application Center

Folder	Permissions Required
"%SystemDrive%\Program Files\Microsoft Application Center	ACA_<computername>: Full Control ACL_<computername>: Read and Execute, List Folder Contents, and Read

The above permissions are required to be able to administer Application Center through the Application Center MMC snap-in. The Member Server Baseline Policy sets the following permissions on the Program Files folder:

Table 12.5: Program Files folder permissions

Folders Secured	Permissions Applied
%systemdrive%\Program Files	Administrators: Full control System: Full control Creator Owner: Full control Users: Read and Execute, List Folder Contents, and Read

In the Member Server Baseline Policy we also configure the option to "Replace existing permissions on all subfolders and files with inheritable permissions." This will overwrite the permissions for the Application Center user accounts, so to avoid this, we configure %SystemDrive%\Program Files\Microsoft Application Center to not allow permissions on the folder to be replaced.

FMStocks ASP.NET Configuration Files

In the FMStocks application, ASP.NET runs in the Presentation tier, but not in the Business tier. There is therefore only a need to configure machine.config on the Presentation tier servers. Also, as FMStocks is the only application being configured, all the appropriate settings can be placed in machine.config and there is no need for a web.config in this case.

In this section we show the changes made to the default machine.config to support FMStocks.

<machine key>

FMStocks runs a Web farm on the Presentation tier, so the validation and decryption keys cannot be automatically generated. Instead we use HashConfigVB.exe to create the keys. This setting should be the same in machine.config for all of the servers in the Presentation tier:

```
<machineKey validationKey="B3FD2180E3CB712347B69DE9689101D68A55D105347AD3B5F7DDA77
3E18736546D914159DE43D2B051B0FB54CECEA6D4F09186F88ABC189DEB3D34DF8B2F8439"
decryptionKey="EA1BB4EA9C39BDAB4A600EF37112E42732A9E39EDCFEF591"
validation="SHA1"/>
```

Note: For more information on HashConfigVB.exe see Knowledge Base articles Q313091, "HOW TO: Create Keys by Using Visual Basic .NET for Use in Forms Authentication" or Q312906, "HOW TO: Create Keys by Using Visual C# .NET for Use in Forms Authentication."

<httpHandlers>

FMStocks does not use Web forms or .NET remoting, so we can disable requests for these types of pages:

```
<httpHandlers>
<add verb="*" path="*.asmx" type="System.Web.HttpForbiddenHandler"/>
<add verb="*" path="*.soap" type="System.Web.HttpForbiddenHandler"/>
<add verb="*" path="*.rem" type="System.Web.HttpForbiddenHandler"/>
...
</httpHandlers>
```

<httpRuntime>

There is no need for a user of FMStocks to upload any information greater than 1Mb (1024K) so to protect against denial-of-service attacks, we prevent this in the machine.config file:

```
<httpRuntime executionTimeout="90"
maxRequestLength="1024"
useFullyQualifiedRedirectUrl="false"
minFreeThreads="8"
minLocalRequestFreeThreads="4"
appRequestQueueLimit="100" />
```

<CustomErrors>

When the FMStocks application is in production, there is no need for users to see detailed error pages that may reveal important information about the application. Instead we have created a custom error page—FMError.htm, and specified that this will be displayed whenever an error occurs:

```
<location path = "FMStocks7">
   <system.web>
      <customErrors defaultRedirect= "FMError.htm " mode="On" />
      <error statusCode= "404 " redirect= "err404FMS7.htm "/>
      <error statusCode= "407 " redirect= "err407FMS7.htm "/>
      </customErrors>
   </system.web>
</location>
```

<WebServices>

FMStocks does not use Simple Object Access Protocol (SOAP), so we explicitly deny the use of the HTTP Soap Protocol:

```
<webServices>
   <protocols>
   <!–FMStocks does not use SOAP, so we'll make it unavailable.–>
      <!–<add name=”HttpSoap”/> –>
        </protocols>
</webServices>
```

<trace>

As the FMStocks application is running in a production environment, we do not want trace information to be available. However we also set the **localonly** attribute, in case the **trace** attribute were changed later for any reason.

```
<location path="Default Web Site/FMStocks7">
   <system.web>
   <trace enabled="false" requestLimit="1" pageOutput="false" localOnly="true"/>

   </system.web>
</location>
```

IIS Lockdown and URLScan

FMStocks can use the URLScan and IISLockdown settings documented in Chapter 10, "Securing .NET Framework Servers Based on Role." You do not need to change any settings to allow the application to function properly on the Presentation tier. The Presentation tier servers should have the .NET Framework Production Server role applied when URLScan is run.

For the Business tier servers we disabled the additional file types: aspx, .ashx, .asmx, .axd, .rem, and .soap. To do this, we created another role in the IISlockd.ini file called FMStocks Business tier and created a new urlscan.ini file called Urlscan_FMstocksBT.ini. In the IISlockd.ini file, the only change from the .NET Framework Production Server role is changing to the new URLScan.ini file. In the URLScan ini file, the only change from the urlscannetprod.ini is the addition of the aspx, .ashx, .asmx, .axd, .rem, and .soap to the **[DenyExtensions]** section of the file.

Securing Network Communications for FMStocks

As well as locking down the servers involved in providing the functionality of FMStocks, we need to secure the communication between them.

The diagram on the next page shows the network environment used for FMStocks and the traffic that needs to be secured.

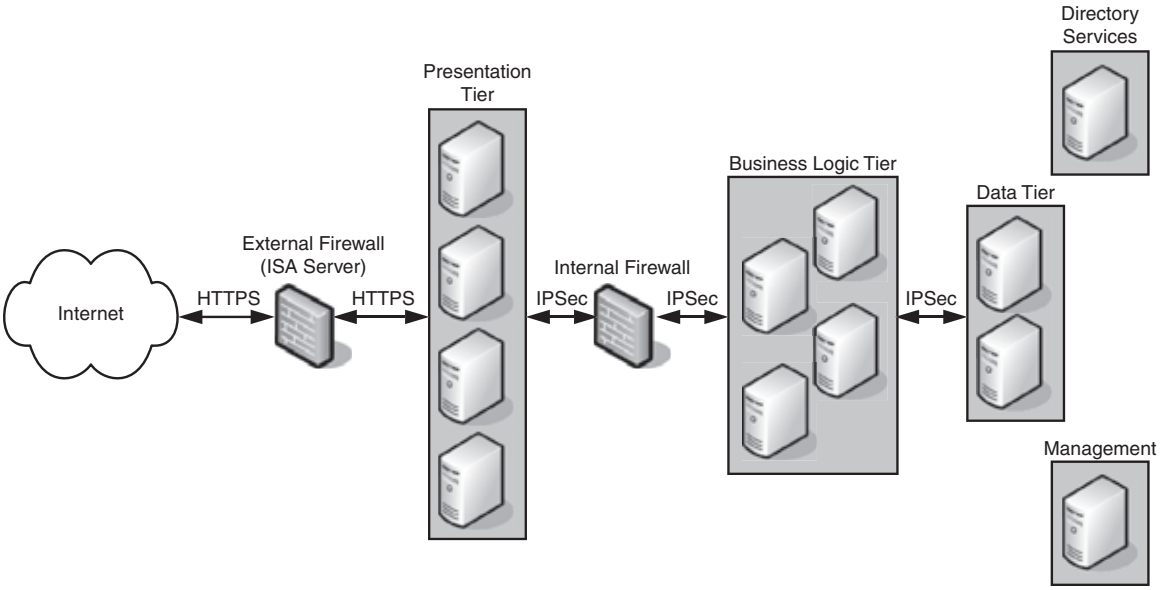


Figure 12.3
FMStocks network environment

To ensure that end-to-end encryption takes place between client computers located anywhere on the Internet and the Presentation tier, we enable Web Publishing for SSL at the ISA server, where incoming SSL requests are redirected to the Presentation tier in the perimeter network as SSL requests. This also allows for content inspection by using the URLScan ISAPI filter at the ISA server. The filter ensures that only approved HTTP verbs and scripting extensions are permitted when connecting to the Presentation tier servers from the Internet.

To enable SSL at both the ISA server and the Presentation tier Web servers, we install the following:

- Certificates from a commercial CA at the ISA server
- Certificates from a private CA at the Web servers in the perimeter network hosting the Presentation tier.

By implementing a certificate issued by a commercial CA at the ISA server, we allow communication from clients without having to install certificates at the client level. The SSL connection between the ISA server and the Presentation tier is a separate SSL session, and only needs the certificates installed at the Presentation tier to be recognized by the ISA server.

We also ensure that the Certificate Distribution Point (CDP) and AIA extensions for issued certificate reference a location that is accessible from both the ISA server and the Presentation tier servers. We do this by publishing the private CA's CRL and CA certificate at an HTTP location that is accessible from the perimeter network.

To ensure all communications between the Presentation tier servers and the Business tier servers are encrypted, we configure them to require IPSec with ESP (3DES and SHA1) for all communications with each other. This is possible because in our environment the internal firewall does not perform NAT. The Presentation tier servers in our environment are not part of the Active Directory domain, so we use certificates for authentication purposes.

We also require IPSec with AH for communications between the Business tier and the Data tier. For authentication, we implement Kerberos authentication, as all Business tier and Data tier servers are members of the Active Directory domain. In fact, by using Kerberos, we ensure that only forest members can connect to the Business tier servers, with no additional configuration required.

Note: To define IPSec filters for communications between servers in a Web cluster, you need to use the individual node addresses of clustered servers, not the shared cluster IP address. The IPSec security associations are established between the individual computers, not the Web clusters. These filters must define the communications that *could* take place between any node in any cluster that exists. For example, if you had three Application Center servers connecting to a back-end cluster SQL database with two nodes, you must define connections from each Application Center node to each SQL back-end database node.

At the domain level, we apply the Client (Respond Only) IPSec policy so that all other computers in the domain can communicate with the Business tier servers by implementing the required IPSec settings. The clients will respond with IPSec if requested.

Our security policy determines that it is not necessary to secure the data to or from Management and Directory Services, as this data is not secured across the entire enterprise. Also the policy does not require that we implement the combination of ESP and AH for our IPSec security associations.

Note: If you implement a combination of both ESP and AH for your IPSec security associations, you must modify the Client (Respond Only) IPSec policy. By default, this policy supports all forms of ESP and AH, but not a security association that implements both IPSec protocols.

Summary

If you follow the steps listed in the previous chapters of this book, you will increase the security of your environment, but at the risk of losing functionality in some of your key applications. In this chapter, we took a specific application and modified the approach to allow that application to run. You should use the techniques outlined in this chapter with your own applications to allow them to function properly in an environment with higher security.

More Information

For the complete *Security Guide to Microsoft Windows 2000 Server*:

http://www.microsoft.com/technet/security/prodtech/windows/windows2000/staysecure/DEFAULT.asp

Details on enabling success auditing for logon events filling the security log:

http://support.microsoft.com/default.aspx?scid=kb;en-us;Q316685

To obtain the IIS Lockdown tool:

http://www.microsoft.com/technet/security/tools/tools/locktool.asp

Details on troubleshooting and configuring IIS Lockdown and URLScan:

http://support.microsoft.com/default.aspx?scid=kb;en-us;Q309677

Further information on SDDL:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/Security/security_descriptor_string_format.asp

For information on building Secure ASP.NET Applications, see Authentication in ASP.NET: .NET Security Guidance:

http://msdn.microsoft.com/practices/

Q313091 HOW TO: Create Keys by Using Visual Basic .NET for Use in Forms Authentication

http://support.microsoft.com/default.aspx?scid=kb;en-us;313091

Q312906 HOW TO: Create Keys by Using Visual C# .NET for Use in Forms Authentication

http://support.microsoft.com/default.aspx?scid=kb;en-us;312906

Appendix A

Files Secured

Files secured by the Member Server Baseline Policy, in addition to the access control lists provided with the hisecws.inf template.

File	Baseline Permissions
%SystemDrive%\Boot.ini	Administrators: Full control System: Full control
%SystemDrive%\Ntdetect.com	Administrators: Full control System: Full control
%SystemDrive%\Ntldr	Administrators: Full control System: Full control
%SystemDrive%\Io.sys	Administrators: Full control System: Full control
%SystemDrive%\Autoexec.bat	Administrators: Full control System: Full control Authenticated Users: Read and Execute, List Folder Contents, and Read
%SystemDrive%\Config.sys	Administrators: Full control System: Full control Authenticated Users: Read and Execute, List Folder Contents, and Read
%SystemRoot%\system32\Append.exe	Administrators: Full control
%SystemRoot%\system32\Arp.exe	Administrators: Full control
%SystemRoot%\system32\At.exe	Administrators: Full control
%SystemRoot%\system32\Attrib.exe	Administrators: Full control

(continued)

File	Baseline Permissions
%SystemRoot%\system32\Cacls.exe	Administrators: Full control
%SystemRoot%\system32\Change.exe	Administrators: Full control
%SystemRoot%\system32\Chcp.com	Administrators: Full control
%SystemRoot%\system32\Chglogon.exe	Administrators: Full control
%SystemRoot%\system32\Chgport.exe	Administrators: Full control
%SystemRoot%\system32\Chguser.exe	Administrators: Full control
%SystemRoot%\system32\Chkdsk.exe	Administrators: Full control
%SystemRoot%\system32\Chkntfs.exe	Administrators: Full control
%SystemRoot%\system32\Cipher.exe	Administrators: Full control
%SystemRoot%\system32\Cluster.exe	Administrators: Full control
%SystemRoot%\system32\Cmd.exe	Administrators: Full control
%SystemRoot%\system32\Compact.exe	Administrators: Full control
%SystemRoot%\system32\Command.com	Administrators: Full control
%SystemRoot%\system32\Convert.exe	Administrators: Full control
%SystemRoot%\system32\Cscript.exe	Administrators: Full control
%SystemRoot%\system32\Debug.exe	Administrators: Full control
%SystemRoot%\system32\Dfscmd.exe	Administrators: Full control
%SystemRoot%\system32\Diskcomp.com	Administrators: Full control
%SystemRoot%\system32\Diskcopy.com	Administrators: Full control
%SystemRoot%\system32\Doskey.exe	Administrators: Full control
%SystemRoot%\system32\Edlin.exe	Administrators: Full control
%SystemRoot%\system32\Exe2bin.exe	Administrators: Full control
%SystemRoot%\system32\Expand.exe	Administrators: Full control
%SystemRoot%\system32\Fc.exe	Administrators: Full control
%SystemRoot%\system32\Find.exe	Administrators: Full control
%SystemRoot%\system32\Findstr.exe	Administrators: Full control
%SystemRoot%\system32\Finger.exe	Administrators: Full control
%SystemRoot%\system32\Forcedos.exe	Administrators: Full control
%SystemRoot%\system32\Format.com	Administrators: Full control
%SystemRoot%\system32\Ftp.exe	Administrators: Full control
%SystemRoot%\system32\Hostname.exe	Administrators: Full control

File	Baseline Permissions
%SystemRoot%\system32\lisreset.exe	Administrators: Full control
%SystemRoot%\system32\Ipconfig.exe	Administrators: Full control
%SystemRoot%\system32\Ipxroute.exe	Administrators: Full control
%SystemRoot%\system32\Label.exe	Administrators: Full control
%SystemRoot%\system32\Logoff.exe	Administrators: Full control
%SystemRoot%\system32\Lpq.exe	Administrators: Full control
%SystemRoot%\system32\Lpr.exe	Administrators: Full control
%SystemRoot%\system32\Makecab.exe	Administrators: Full control
%SystemRoot%\system32\Mem.exe	Administrators: Full control
%SystemRoot%\system32\Mmc.exe	Administrators: Full control
%SystemRoot%\system32\Mode.com	Administrators: Full control
%SystemRoot%\system32\More.com	Administrators: Full control
%SystemRoot%\system32\Mountvol.exe	Administrators: Full control
%SystemRoot%\system32\Msg.exe	Administrators: Full control
%SystemRoot%\system32\Nbtstat.exe	Administrators: Full control
%SystemRoot%\system32\Net.exe	Administrators: Full control
%SystemRoot%\system32\Net1.exe	Administrators: Full control
%SystemRoot%\system32\Netsh.exe	Administrators: Full control
%SystemRoot%\system32\Netstat.exe	Administrators: Full control
%SystemRoot%\system32\Nslookup.exe	Administrators: Full control
%SystemRoot%\system32\Ntbackup.exe	Administrators: Full control
%SystemRoot%\system32\Ntsd.exe	Administrators: Full control
%SystemRoot%\system32\Pathping.exe	Administrators: Full control
%SystemRoot%\system32\Ping.exe	Administrators: Full control
%SystemRoot%\system32\Print.exe	Administrators: Full control
%SystemRoot%\system32\Query.exe	Administrators: Full control
%SystemRoot%\system32\Rasdial.exe	Administrators: Full control
%SystemRoot%\system32\Rcp.exe	Administrators: Full control
%SystemRoot%\system32\Recover.exe	Administrators: Full control
%SystemRoot%\system32\Regedit.exe	Administrators: Full control

(continued)

File	Baseline Permissions
%SystemRoot%\system32\Regedt32.exe	Administrators: Full control
%SystemRoot%\system32\Regini.exe	Administrators: Full control
%SystemRoot%\system32\Register.exe	Administrators: Full control
%SystemRoot%\system32\Regsvr32.exe	Administrators: Full control
%SystemRoot%\system32\Replace.exe	Administrators: Full control
%SystemRoot%\system32\Reset.exe	Administrators: Full control
%SystemRoot%\system32\Rexec.exe	Administrators: Full control
%SystemRoot%\system32\Route.exe	Administrators: Full control
%SystemRoot%\system32\Routemon.exe	Administrators: Full control
%SystemRoot%\system32\Router.exe	Administrators: Full control
%SystemRoot%\system32\Rsh.exe	Administrators: Full control
%SystemRoot%\system32\Runas.exe	Administrators: Full control
%SystemRoot%\system32\Runonce.exe	Administrators: Full control
%SystemRoot%\system32\Secedit.exe	Administrators: Full control
%SystemRoot%\system32\Setpwd.exe	Administrators: Full control
%SystemRoot%\system32\Shadow.exe	Administrators: Full control
%SystemRoot%\system32\Share.exe	Administrators: Full control
%SystemRoot%\system32\Snmp.exe	Administrators: Full control
%SystemRoot%\system32\Snmptrap.exe	Administrators: Full control
%SystemRoot%\system32\Subst.exe	Administrators: Full control
%SystemRoot%\system32\Telnet.exe	Administrators: Full control
%SystemRoot%\system32\Termsrv.exe	Administrators: Full control
%SystemRoot%\system32\Tftp.exe	Administrators: Full control
%SystemRoot%\system32\TIntadmn.exe	Administrators: Full control
%SystemRoot%\system32\TIntsess.exe	Administrators: Full control
%SystemRoot%\system32\TIntsvr.exe	Administrators: Full control
%SystemRoot%\system32\Tracert.exe	Administrators: Full control
%SystemRoot%\system32\Tree.com	Administrators: Full control
%SystemRoot%\system32\Tsadmin.exe	Administrators: Full control
%SystemRoot%\system32\Tscon.exe	Administrators: Full control
%SystemRoot%\system32\Tsdiscon.exe	Administrators: Full control

File	Baseline Permissions
%SystemRoot%\system32\Tskill.exe	Administrators: Full control
%SystemRoot%\system32\Tsprof.exe	Administrators: Full control
%SystemRoot%\system32\Tsshutdn.exe	Administrators: Full control
%SystemRoot%\system32\Usrmgr.com	Administrators: Full control
%SystemRoot%\system32\Wscript.exe	Administrators: Full control
%SystemRoot%\system32\Xcopy.exe	Administrators: Full control

Appendix B

Default Windows 2000 Services

The Default column shows the service startup for a Windows 2000-based server. The Baseline column shows the configure startup for each service after the Member Server Baseline Policy is applied.

Service	Full Name	Default	Baseline
Alerter	Alerter	Automatic	Disabled
AppMgmt	Application Management	Manual	Disabled
ClipSrv	ClipBook	Manual	Disabled
EventSystem	COM+ Event System	Manual	Manual
Browser	Computer Browser	Automatic	Disabled
DHCP	DHCP Client	Automatic	Automatic
Dfs	Distributed File System	Automatic	Enabled only in the DC role
TrkWks	Distributed Link Tracking Client	Automatic	Automatic
TrkSrv	Distributed Link Tracking Server	Manual	Disabled
MSDTC	Distributed Transaction Coordinator	Automatic	Disabled
DNSCache	DNS Client	Automatic	Automatic
EventLog	Event Log	Automatic	Automatic
Fax	Fax Service	Manual	Disabled
NtFrs	File Replication	Manual	Disabled
IISADMIN	IIS Admin Service	Automatic	Disabled
Cisvc	Indexing Service	Manual	Disabled

(continued)

Service	Full Name	Default	Baseline
SharedAccess	Internet Connection Sharing	Manual	Disabled
IsmServ	Intersite Messaging	Disabled	Disabled
PolicyAgent	IPSEC Policy Agent(IPSEC Service)	Automatic	Disabled
Kdc	Kerberos Key Distribution Center	Disabled	Enabled only in the DC role
LicenseService	License Logging Service	Automatic	Disabled
Dmserver	Logical Disk Manager	Automatic	Automatic
Dmadmin	Logical Disk Manager Administrative Service	Manual	Manual
Messenger	Messenger	Automatic	Disabled
Netlogon	Net Logon	Automatic*	Automatic
Mnmsrvc	NetMeeting Remote Desktop Sharing	Manual	Disabled
Netman	Network Connections	Manual	Manual
NetDDE	Network DDE	Manual	Disabled
NetDDEdsdm	Network DDE DSDM	Manual	Disabled
NtLmSsp	NTLM Security Support Provider	Manual	Disabled
SysmonLog	Performance Logs and Alerts	Manual	Manual
PlugPLay	Plug and Play	Automatic	Automatic
Spooler	Print Spooler	Automatic	Disabled
ProtectedStorage	Protected Storage	Automatic	Automatic
RSVP	QoS RSVP	Manual	Disabled
RasAuto	Remote Access Auto Connection Manager	Manual	Disabled
RasMan	Remote Access Connection Manager	Manual	Disabled
RpcSs	Remote Procedure Call (RPC)	Automatic	Automatic
Rpclocator	Remote Procedure Call (RPC) Locator	Manual	Enabled only in the DC role
RemoteRegistry	Remote Registry Service	Automatic	Automatic
NtmsSvc	Removable Storage	Automatic	Disabled
RemoteAccess	Routing and Remote Access	Disabled	Disabled
Seclogon	RunAs Service	Automatic	Disabled

Service	Full Name	Default	Baseline
SamSs	Security Accounts Manager	Automatic	Automatic
Lanmanserver	Server	Automatic	Automatic
SMTPSVC	Simple Mail Transport Protocol (SMTP)	Automatic	Disabled
ScardSvr	Smart Card	Manual	Disabled
ScardDrv	Smart Card Helper	Manual	Disabled
SENS	System Event Notification	Automatic	Automatic
Schedule	Task Scheduler	Automatic	Disabled
LmHosts	TCP/IP NetBIOS Helper Service	Automatic	Automatic
TapiSrv	Telephony	Manual	Disabled
TlntSvr	Telnet	Manual	Disabled
TermService	Terminal Services	Disabled	Disabled
UPS	Uninterruptible Power Supply	Manual	Disabled
UtilMan	Utility Manager	Manual	Disabled
MSIServer	Windows Installer	Manual	Disabled
WinMgmt	Windows Management Instrumentation	Manual	Disabled
WMI	Windows Management Instrumentation Driver Extensions	Manual	Manual
W32Time	Windows Time	Automatic*	Automatic
LanmanWorkstation	WorkStation	Automatic	Automatic
W3svc	World Wide Web Publishing Service	Automatic	Disabled

* - Automatic for a server in the domain. Manual if server belongs to a workgroup.

Appendix C

Additional Services

The following table lists additional services that are included with Windows 2000 Server and Advanced Server and can be added to a default installation.

Service	Full Name	Baseline
BINLSVC	Boot Information Negotiation Layer	Disabled
CertSvc	Certificate Services	Disabled
ClusSvc	Cluster Service	Disabled
DHCPServer	DHCP Server	Disabled
DNS	DNS Server	Enabled only in the DC role
MacFile	File Server for Macintosh	Disabled
MSFTPSVC	FTP Publishing Service	Disabled
NWCWorkstation	Gateway Service for Netware	Disabled
IAS	Internet Authentication Service	Disabled
MSMQ	Message Queuing	Disabled
NntpSvc	Network News Transport Protocol (NNTP)	Disabled
NSLService	On-Line Presentation Broadcast	Disabled
MacPrint	Print Server for Macintosh	Disabled
RSVP	QoS RSVP	Disabled
Remote_Storage_Engine	Remote Storage Engine	Disabled
Remote_Storage_File_System_Agent	Remote Storage File	Disabled

(continued)

Service	Full Name	Baseline
Remote_Storage_Subsystem	Remote Storage Media	Disabled
Remote_Storage_User_Link	Remote Storage Notification	Disabled
NwSapAgent	SAP Agent	Disabled
SimpTcp	Simple TCP/IP Services	Disabled
Groveler	Single Instance Storage Groveler	Disabled
LDAPSVCX	Site Server ILS Service	Disabled
SNMP	SNMP Service	Disabled
SNMPTRAP	SNMP Trap Service	Disabled
LPDSVC	TCP/IP Print Server	Disabled
TermServLicensing	Terminal Services Licensing	Disabled
TFTPD	Trivial FTP Daemon	Disabled
WINS	Windows Internet Name Service (WINS)	Disabled
nsmonitor	Windows Media Monitor Service	Disabled
nsprogram	Windows Media Program Service	Disabled
nsstation	Windows Media Station Service	Disabled
nsunicast	Windows Media Unicast Service	Disabled

Appendix D

IIS Security Settings

This appendix contains the Iislockd.ini file sections for the .NET Framework Production Server and .NET Framework Development Server roles. See Chapter 4: "Securing Servers Running .Net-based Applications" for instructions on adding these to the Iislockd.ini file.

.NET Framework Production Server

```
[netProd]
label=".NET Framework Production Server (Debug disabled)"

Enable_iis_http=TRUE
Enable_iis_ftp= FALSE
Enable_iis_smtp= FALSE
Enable_iis_nntp= FALSE
Enable_asp= TRUE
Enable_index_server_web_interface= FALSE
Enable_server_side_includes= FALSE
Enable_internet_data_connector= FALSE
Enable_internet_printing= FALSE
Enable_HTR_scripting= FALSE
Enable_webDAV= FALSE
Disable_Anonymous_user_system_utility_execute_rights= TRUE
Disable_Anonymous_user_content_directory_write_rights= TRUE
Remove_iissamples_virtual_directory=TRUE
Remove_scripts_directory=TRUE
Remove_MSADC_virtual_directory=TRUE
Remove_iisadmin_virtual_directory=TRUE
Remove_iishelp_virtual_directory=TRUE
UrlScan_Install=TRUE
UrlScan_IniFileLocation=urlscan_netProd.ini
AdvancedSetup =
UninstallServices=FALSE
```

.NET Framework Development Server

```
 [netDev]
label=".NET Framework Development Server"

Enable_iis_http=TRUE
Enable_iis_ftp= FALSE
Enable_iis_smtp= FALSE
Enable_iis_nntp= FALSE
Enable_asp= TRUE
Enable_index_server_web_interface= FALSE
Enable_server_side_includes= FALSE
Enable_internet_data_connector= FALSE
Enable_internet_printing= FALSE
Enable_HTR_scripting= FALSE
Enable_webDAV= FALSE
Disable_Anonymous_user_system_utility_execute_rights= TRUE
Disable_Anonymous_user_content_directory_write_rights= TRUE
Remove_iissamples_virtual_directory=TRUE
Remove_scripts_directory=TRUE
Remove_MSADC_virtual_directory=TRUE
Remove_iisadmin_virtual_directory=TRUE
Remove_iishelp_virtual_directory=TRUE
UrlScan_Install=TRUE
UrlScan_IniFileLocation=urlscan_netDev.ini
AdvancedSetup =
UninstallServices=FALSE
```

Microsoft®

patterns & practices

Proven practices for predictable results

Patterns & practices are Microsoft's recommendations for architects, software developers, and IT professionals responsible for delivering and managing enterprise systems on the Microsoft platform. Patterns & practices are available for both IT infrastructure and software development topics.

Patterns & practices are based on real-world experiences that go far beyond white papers to help enterprise IT pros and developers quickly deliver sound solutions. This technical guidance is reviewed and approved by Microsoft engineering teams, consultants, Product Support Services, and by partners and customers. Organizations around the world have used patterns & practices to:

Reduce project cost

- Exploit Microsoft's engineering efforts to save time and money on projects
- Follow Microsoft's recommendations to lower project risks and achieve predictable outcomes

Increase confidence in solutions

- Build solutions on Microsoft's proven recommendations for total confidence and predictable results
- Provide guidance that is thoroughly tested and supported by PSS, not just samples, but production quality recommendations and code

Deliver strategic IT advantage

- Gain practical advice for solving business and IT problems today, while preparing companies to take full advantage of future Microsoft technologies.

To learn more about *patterns & practices* visit: *msdn.microsoft.com/practices*

To purchase *patterns & practices* guides visit: *shop.microsoft.com/practices*

patterns & practices
Proven practices for predictable results

patterns & practices

Proven practices for predictable results

Patterns & practices are available for both IT infrastructure and software development topics. There are four types of patterns & practices available:

Reference Architectures

Reference Architectures are IT system-level architectures that address the business requirements, operational requirements, and technical constraints for commonly occurring scenarios. Reference Architectures focus on planning the architecture of IT systems and are most useful for architects.

Reference Building Blocks

References Building Blocks are re-usable sub-systems designs that address common technical challenges across a wide range of scenarios. Many include tested reference implementations to accelerate development.

Reference Building Blocks focus on the design and implementation of sub-systems and are most useful for designers and implementors.

Operational Practices

Operational Practices provide guidance for deploying and managing solutions in a production environment and are based on the Microsoft Operations Framework. Operational Practices focus on critical tasks and procedures and are most useful for production support personnel.

Patterns

Patterns are documented proven practices that enable re-use of experience gained from solving similar problems in the past. Patterns are useful to anyone responsible for determining the approach to architecture, design, implementation, or operations problems.

To learn more about *patterns & practices* visit: *msdn.microsoft.com/practices*

To purchase *patterns & practices* guides visit: *shop.microsoft.com/practices*

patterns & practices current titles

December 2002

Reference Architectures

Microsoft Systems Architecture—Enterprise Data Center *2007 pages*
Microsoft Systems Architecture—Internet Data Center *397 pages*
Application Architecture for .NET: Designing Applications and Services *127 pages*
Microsoft SQL Server 2000 High Availability Series: Volume 1: Planning *92 pages*
Microsoft SQL Server 2000 High Availability Series: Volume 2: Deployment *128 pages*
Enterprise Notification Reference Architecture for Exchange 2000 Server *224 pages*
Microsoft Content Integration Pack for Content Management Server 2001 and SharePoint Portal Server 2001 *124 pages*
UNIX Application Migration Guide *694 pages*
Microsoft Active Directory Branch Office Guide: Volume 1: Planning *88 pages*
Microsoft Active Directory Branch Office Series Volume 2: Deployment and Operations *195 pages*
Microsoft Exchange 2000 Server Hosting Series Volume 1: Planning *227 pages*
Microsoft Exchange 2000 Server Hosting Series Volume 2: Deployment *135 pages*
Microsoft Exchange 2000 Server Upgrade Series Volume 1: Planning *306 pages*
Microsoft Exchange 2000 Server Upgrade Series Volume 2: Deployment *166 pages*

Reference Building Blocks

Data Access Application Block for .NET *279 pages*
.NET Data Access Architecture Guide *60 pages*
Designing Data Tier Components and Passing Data Through Tiers *70 pages*
Exception Management Application Block for .NET *307 pages*
Exception Management in .NET *35 pages*
Monitoring in .NET Distributed Application Design *40 pages*
Microsoft .NET/COM Migration and Interoperability *35 pages*
Production Debugging for .NET-Connected Applications *176 pages*
Authentication in ASP.NET: .NET Security Guidance *58 pages*
Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication *608 pages*

Operational Practices

Security Operations Guide for Exchange 2000 Server *136 pages*
Security Operations for Microsoft Windows 2000 Server *188 pages*
Microsoft Exchange 2000 Server Operations Guide *113 pages*
Microsoft SQL Server 2000 Operations Guide *170 pages*
Deploying .NET Applications: Lifecycle Guide *142 pages*
Team Development with Visual Studio .NET and Visual SourceSafe *74 pages*
Backup and Restore for Internet Data Center *294 pages*

For current list of titles visit: *msdn.microsoft.com/practices*

To purchase *patterns & practices* guides visit: *shop.microsoft.com/practices*

patterns & practices
Proven practices for predictable results