Microsoft

2
SECOND EDITION

Windows® Command-Line

William R. Stanek
Author and Series Editor

Administrator's Pocket Consultant

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Control Number: 2008927283

Printed and bound in the United States of America.

1 2 3 4 5 6 7 8 9 QWE 3 2 1 0 9 8

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to mspinput@microsoft.com.

Acquisitions Editor: Martin DelRe
Developmental Editor: Karen Szall
Project Editor: Maria Gargiulo
Editorial Production: ICC Macmillan, Inc.
Technical Reviewer: James Johnson, Technical Review services provided by Content Master, a member of CM Group, Ltd
Cover: Tom Draper Design

Body Part No. X14-71538

Contents at a Glance

Table of Contents

List of Tables

Acknowledgments

Wanting to do something fundamentally different from how it's been done before turned out to be much harder than I ever thought—and, I hope, completely rewarding for you. You see, there are plenty of books for Windows administrators—and plenty of books for people who want to script Windows—but no one had sat down before and written an entire book on Windows administration from the command line that was truly focused on administration and not on the commands themselves. I hope that the result of all the hard work is that the book you hold in your hands is something unique. This isn't the kind of command-line book that says, "Here's the Edit command. You use this command to do this and this, and here are its parameters." Sure, some of that is included—as in any book for administrators—but this book focuses on using the command line in the context of everyday administration. It teaches you how to perform the daily administrative procedures and details how those procedures are implemented from the command line. So whether you want to learn how to manage daily operations, track Windows performance, view the event logs, partition disks, configure TCP/IP, or perform hundreds of others tasks, this book has the answers as they relate to the command line.

As I've stated in *Windows Server 2008 Administrator's Pocket Consultant* (Microsoft Press 2008) and in *Internet Information Services (IIS) 7.0 Administrator's Pocket Consultant* (Microsoft Press, 2008), the team at Microsoft Press is top-notch. On this project, I worked with Karen Szall, Devon Musgrave, Maria Gargiulo, and others at Microsoft. They were all very helpful throughout the writing process. Thanks also to Martin DelRe for believing in my work and shepherding it through production.

Unfortunately for the writer (but fortunately for readers), writing is only one part of the publishing process. Next came editing and author review. I must say, Microsoft Press has the most thorough editorial and technical review process I've seen anywhere—and I've written a lot of books for many different publishers. John Pierce was the project editor for the book and helped work the book through the editorial process. This was our first time working together, and it was a good experience. Jim Johnson was the technical editor for the book. Jim was also the technical editor for the first edition of the book, and it was good to work with him again. Becka McKay was the book's copy editor.

I would also like to extend a personal thanks to Lucinda Rowley, Anne Hamilton, and Chris Nelson. They've helped out at many points of my writing career and been there when I needed them the most. Thank you!

I hope I haven't forgotten anyone, but if I have, it was an oversight. *Honest.* ;-)

Introduction

Windows Command-Line Administrator's Pocket Consultant, Second Edition, is designed to be a concise and compulsively usable resource for Windows administrators. This is the readable resource guide that you'll want on your desk or in your pocket at all times. The book discusses everything you need to perform core administrative tasks using the Windows command line. Because the focus is directed toward providing you with maximum value in a pocket-sized guide, you don't have to wade through hundreds of pages of extraneous information to find what you're looking for. Instead, you'll find exactly what you need to get the job done.

In short, the book is designed to be the one resource you consult whenever you have questions regarding Windows command-line administration. To this end, the book concentrates on daily administration procedures, frequently used tasks, documented examples, and options that are representative but not necessarily inclusive. The goal is to keep the content so concise that the book remains compact and easy to navigate, while ensuring that the book is packed with as much information as possible—making it a valuable resource. Thus, instead of a hefty 1,000-page tome or a lightweight 100-page quick reference, you get a valuable resource guide that can help you quickly and easily perform common tasks, solve problems, and implement such advanced administration areas as automated monitoring, memory leak analysis, disk partitioning, Active Directory management, and network troubleshooting.

Find Additional Content Online As new or updated material becomes available that complements your book, it will be posted online on the Microsoft Press Online Windows Server and Client Web site. Based on the final build of Windows Server 2008, the type of material you might find includes updates to book content, articles, links to companion content, errata, sample chapters, and more. This Web site will be available soon at *www.microsoft.com/learning/books/online/serverclient*, and will be updated periodically.

Who Is This Book For?

Windows Command-Line Administrator's Pocket Consultant, Second Edition, covers Windows Server 2008 and Windows Vista. The book is designed for:

- Current Windows Server 2008 administrators
- Support staff who maintain Windows Vista systems
- Accomplished users who have some administrator responsibilities
- Administrators upgrading to Windows Server from previous versions
- Administrators transferring from other platforms

To pack in as much information as possible, I had to assume that you have basic networking skills and a basic understanding of Windows, and that Windows is already installed on your systems. With this in mind, I don't devote entire chapters to understanding Windows architecture, installing Windows, or Windows startup and shutdown. I do, however, cover scheduling tasks, monitoring Windows systems, managing accounts, administering network services, and much more.

I also assume that you are fairly familiar with Windows commands and procedures as well as the Windows user interface. If you need help learning Windows basics, you should read the Windows documentation.

How Is This Book Organized?

Windows Command-Line Administrator's Pocket Consultant, Second Edition, is designed to be used in the daily administration of Windows systems, and as such the book is organized by job-related tasks rather than by Windows features. Speed and ease of reference is an essential part of this hands-on guide. The book has an expanded table of contents and an extensive index for finding answers to problems quickly. Many other quick reference features have been added as well. These features include quick step-by-step instructions, lists, tables with fast facts, and extensive cross-references. The book is organized into both parts and chapters.

Part I, "Windows Command-Line Fundamentals," reviews the fundamental tasks you need for command-line administration. Chapter 1 provides an overview of command-line administration tools, techniques, and concepts. Chapter 2 is designed to help you get the most out of the command shell. It details techniques for starting up the command shell using parameters, how to control command path settings, what redirection techniques are available, and how to use multiple commands in sequences. Chapter 3 discusses the essentials for creating command-line scripts. You'll learn how to set variables, work with conditional controls, and create procedures.

Windows provides many command-line tools to help in the management of daily operations. Part II, "Windows Systems Administration Using the Command Line," discusses the core tools and techniques you'll use to manage Windows systems. Chapter 4 explores techniques for configuring roles, role services, and features on Windows servers. Chapter 5 discusses many of the key administration tools, including those that help you gather system information, work with the Windows registry, configure Windows services, and shut down systems remotely. Chapter 6 examines the logging tools available for Windows systems that can help you identify and track system problems, monitor applications and services, and maintain system security. You'll also learn how to write events to the system and application logs. In Chapter 7, you'll learn about tools and techniques for monitoring applications, examining

processes, and maintaining performance. Chapter 8 provides techniques you can use to manage the way logging is performed, centralize event logging across the enterprise, and collect and generate reports on performance data. Chapter 9 discusses ways you can automate tasks to reduce the daily workload.

The book continues with Part III, "Windows File System and Disk Administration Using the Command Line." Users depend on hard disk drives to store their word-processing documents, spreadsheets, and other types of data. If you've worked with Windows Vista or Windows Server 2008 for any length of time, you've probably used the Disk Management tool. The command-line counterpart of Disk Management is the disk partition utility (DiskPart). You can use DiskPart to handle most disk management tasks as well as to perform some additional tasks that cannot be performed in the graphical user interface. Chapter 10 provides an introduction to DiskPart and also discusses FSUtil, ChkDsk, and CHKNTFS. Chapter 11 discusses partitioning basic disks. Chapter 12 examines dynamic disks and how they are used. The chapter also examines implementing, managing, and troubleshooting RAID.

Part IV, "Windows Active Directory Administration Using the Command Line," concentrates on the core commands you'll use for configuring, managing, and troubleshooting Active Directory. Chapter 13 discusses many of the key directory services administration tools, including tools that help you gather directory information. Chapter 14 examines tools that help you create and manage computer accounts in Active Directory. You'll also learn how to configure domain controllers as global catalogs and operations masters. Chapter 15 discusses creating and managing accounts for users and groups in Active Directory.

The final part, Part V, "Windows Network Administration Using the Command Line," examines network printing, TCP/IP networking, and related issues. Chapter 16 examines network printing and print services. Chapter 17 discusses configuring, maintaining, and troubleshooting TCP/IP networking from the command line.

Appendix A provides a quick reference for command-line utilities discussed in the book. In Appendix B, you'll find a quick reference for the contexts and commands available when you are working with the network services shell (Netsh). You can use Netsh to manage the configuration of various network services on local and remote computers.

Conventions Used in This Book

I've used a variety of elements to help keep the text clear and easy to follow. You'll find code terms and listings in monospace type, except when I tell you to actually type a command. In that case, the command appears in bold type. When I introduce and define a new term, I put it in *italics*.

Other conventions include:

- **Notes** To provide details on a point that needs emphasis
- **Best Practices** To examine the best technique to use when working with advanced configuration and administration concepts
- **Cautions** To warn you when there are potential problems you should look out for
- **More Info** To provide more information on the subject
- **Real World** To provide real-world advice when discussing advanced topics
- **Security Alerts** To point out important security issues
- **Tips** To offer helpful hints or additional information

I truly hope you find that *Windows Command-Line Administrator's Pocket Consultant,* Second Edition, provides everything that you need to perform essential administrative tasks as quickly and efficiently as possible. You're welcome to send your thoughts to me at williamstanek@aol.com. Thank you.

Support

Every effort has been made to ensure the accuracy of this book. Microsoft Press provides corrections for books through the World Wide Web at the following address:

http://www.microsoft.com/mspress/support

If you have comments, questions, or ideas about this book, please send them to Microsoft Press using either of the following methods:

Postal Mail:

Microsoft Press
Attn: Editor, *Windows Command-Line Administrator's Pocket Consultant,* Second Edition
One Microsoft Way
Redmond, WA 98052-6399

E-mail:

mspinput@microsoft.com

Please note that product support isn't offered through these mail addresses. For support information, visit Microsoft's Web site at *http://support.microsoft.com/*.

Chapter 5

Managing Windows Systems

Your job as an administrator is to plan, organize, and track the details that keep the network running. If you're to survive without just muddling through, you need to learn how to do those jobs quickly and efficiently. Fortunately, Windows supplies plenty of command-line tools to help you with these tasks, and this chapter discusses some of the more important tools for daily systems management.

Examining System Information

Often when you are working with a user's computer or a remote server, you'll want to examine some basic system information, such as who is logged on, the current system time, or the location of a certain file. Commands that help you gather basic system information include the following:

- **DATE** Displays and sets the current system date
- **TIME** Displays and sets the current system time
- **WHOAMI** Displays the name of the user currently logged on the system, such as adatum\administrator
- **WHERE** Searches for files using a search pattern and returns a list of matching results

To use DATE or TIME, simply type the command in a command shell window followed by the /T parameter and press Enter. The output of date /t is the current date, such as **Wed 03/19/2008**. The output of time /t is the current time, such as **04:35 PM**. To set the date or time, simply follow the command name with the desired date or time. You enter the current date in MM-DD-YY format, where MM is for the two-digit month, DD is for the two-digit day, and YY is for the two-digit year, such as entering **03-20-08** for March 20, 2008.

You enter the current time in HH:MM or HH:MM:SS format, where HH is for the two-digit hour, MM is for the two-digit minute, and SS is for the two-digit second. If you enter the time without designating A.M. or P.M., the TIME command assumes you are using a 24-hour clock, where hours from 00 to 11 indicate A.M. and hours from 12 to 23 indicate P.M. All of the following examples set the time to 4:45 P.M.:

```
time 04:45 PM
time 04:45:00 PM
time 16:45:00
```

To use WHOAMI to determine who is logged on, simply type the command in a command shell window and press Enter. If the computer is part of a workgroup, the

output includes the name of the computer followed by a backward slash and the name of the logged-on user, such as computer84\deanr. If the computer is part of a domain, the output includes the name of the domain followed by a backward slash and the name of the logged-on user, such as adatum\williams.

By default, WHERE searches the current directory and in the paths specified by the PATH environment variable. This means you can quickly search for an executable in the current path simply by typing **where** followed by the executable you want to find. For example, if you want to find CMD.EXE, you can type the following:

```
where cmd.exe
```

The output is the full file path to CMD.EXE, such as:

```
C:\Windows\System32\cmd.exe
```

With WHERE, the other most common syntax you'll use is

```
where /r baseDir filename
```

Here, /r is for a recursive search starting from the specified directory (*BaseDir*) and including all subdirectories, and *filename* is the name or partial name of the file to search for, which can include wildcard characters. Use ? as a wildcard to match a single character and * as a wildcard to match multiple characters, such as data???.txt or data*.*. In the following example, you search the C:\ directory and all subdirectories for text files that begin with *data*, as follows:

```
where /r C:\ data*.txt
```

You can also search for files of all types that begin with *data*, as in this example:

```
where /r C:\ data*.*
```

Sometimes when you are working with a computer, you'll want to obtain information on the system configuration or the system environment. With mission-critical systems, you may want to save or print this information for easy reference. Commands that help you gather system information include the following:

- **DRIVERQUERY** Displays a list of all installed device drivers and their properties, including module name, display name, driver type, and driver link date. With verbose output, the command also lists the driver status, state, start mode, memory usage, and file system path. Use the /V parameter to get verbose output of all unsigned drivers.
- **SYSTEMINFO** Displays detailed system configuration information, including operating system version, system type, system manufacturer, processor, BIOS version, memory size, locale setting, time zone setting, and network card configuration. This command also shows the hotfixes that have been installed.

To use these commands on a local computer, simply type the command name in a command shell window and press Enter. With DRIVERQUERY, use the /V parameter

to get verbose output for unsigned drivers, and the /Si parameter to display properties of signed drivers, such as

```
driverquery /si
```

With the DRIVERQUERY and SYSTEMINFO commands, you can also specify the remote computer to query and the Run As permissions. To do this, you must use the expanded syntax, which includes the following parameters:

```
/S Computer /U [Domain\]User [/P Password]
```

where *Computer* is the remote computer name or IP address, *Domain* is the optional domain name in which the user account is located, *User* is the name of the user account whose permissions you want to use, and *Password* is the optional password for the user account. If you don't specify the domain, the current domain is assumed. If you don't provide the account password, you are prompted for the password.

To see how the computer and user information can be added to the syntax, consider the following examples:

Use the account adatum\wrstanek when querying MAILER1 for driver settings:

```
driverquery /s mailer1 /u adatum\wrstanek
```

Use the account adatum\administrator when querying CORPSERVER01 for system information:

```
systeminfo /s corpserver01 /u adatum\administrator
```

Tip The basic output of these commands is in table format. You can also format the output as a list or lines of comma-separated values using /Fo List or /Fo Csv, respectively. You may wonder why you should use the various formats. That's a good question. For SYSTEMINFO, I recommend using the list format (/ Fo List) when you want to see all details about system configuration, and for DRIVERQUERY I recommend the verbose list format (/Fo List /V) when you are troubleshooting unsigned drivers. Further, I recommend using comma-separated values when you want to store the output in a file that may later be exported to a spreadsheet or flat-file database. Remember you can redirect the output of the DRIVERQUERY and SYSTEMINFO commands to a file by using output redirection (> or >>).

Working with the Registry

The Windows registry stores configuration settings. Using the Reg command-line utility, you can view, add, delete, compare, and copy registry entries. Because the Windows registry is essential to the proper operation of the operating system, make

changes to the registry only when you know how these changes will affect the system. Before you edit the registry in any way, perform a complete system backup and create a system recovery data snapshot. This way, if you make a mistake, you can recover the registry and the system.

> **Caution** Improperly modifying the Windows registry can cause serious problems. If the registry becomes corrupted, you might have to reinstall the operating system. Double-check the commands you use before executing them. Make sure that they do exactly what you intend.

Understanding Registry Keys and Values

The Windows registry stores configuration settings for the operating system, applications, users, and hardware. Registry settings are stored as keys and values, which are placed under a specific root key controlling when and how the keys and values are used.

Table 5-1 lists the registry root keys as well as a description and the reference name you will use to refer to the root key when working with the REG command. Under the root keys, you'll find the main keys that control system, user, application, and hardware settings. These keys are organized into a tree structure, with folders representing keys. For example, under HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services, you'll find folders for all services installed on the system. Within these folders are the registry keys that store important service configuration settings and their subkeys.

Table 5-1 Keys in the Windows Registry

Root Key	Reference Name	Description
HKEY_CURRENT_USER	HKCU	Stores configuration settings for the current user.
HKEY_LOCAL_MACHINE	HKLM	Stores system-level configuration settings.
HKEY_CLASSES_ROOT	HKCR	Stores configuration settings for applications and files. Also ensures that the correct application is opened when a file is accessed.
HKEY_USERS	HKU	Stores default-user and other-user settings by profile.
HKEY_CURRENT_CONFIG	HKCC	Stores information about the hardware profile being used.

Keys that you want to work with must be designated by their folder path. For example, the path to the DNS key is HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DNS and, using the abbreviated path HKLM\SYSTEM\CurrentControlSet\Services\DNS, you can view and manipulate this key.

Key values are stored as a specific data type. Table 5-2 provides a summary of the main data types used with keys.

Table 5-2 Registry Key Values and Data Types

Data Type	Description	Example
REG_BINARY	Identifies a binary value. Binary values are stored using base-2 (0 or 1 only) but are displayed and entered in hexadecimal (base-16) format.	01 00 14 80 90 00 00 9c 00
REG_DWORD	Identifies a binary data type in which 32-bit integer values are stored as four byte-length values in hexadecimal.	0x00000002
REG_EXPAND_SZ	Identifies an expandable string value, which is usually used with directory paths.	%SystemRoot%\dns.exe
REG_MULTI_SZ	Identifies a multiple string value.	Tcpip Afd RpcSc
REG_NONE	Identifies data without a particular type. This data is written as binary values but displayed and entered in hexadecimal (base-16) format.	23 45 67 80
REG_SZ	Identifies a string value containing a sequence of characters.	DNS Server

As long as you know the key path and understand the available key data types, you can use the REG command to view and manipulate keys in a variety of ways. REG has several different subcommands, and we'll explore some of them. The sections that follow discuss each of the following REG subcommands:

- **REG add** Adds a new subkey or entry to the registry
- **REG delete** Deletes a subkey or entries from the registry
- **REG query** Lists the entries under a key and the names of subkeys (if any)

- **REG compare** Compares registry subkeys or entries
- **REG copy** Copies a registry entry to a specified key path on a local or remote system
- **REG flags** Displays and manages the current flags of a specified key
- **REG restore** Writes saved subkeys, entries, and values back to the registry
- **REG save** Saves a copy of specified subkeys, entries, and values to a file

The following sections will also discuss the following commands for performing advanced registry manipulation:

- **REG import** Imports a specified hive file into the registry
- **REG export** Exports specified subkeys, entries, and values to a registry file
- **REG load** Loads a specified hive file into the registry
- **REG unload** Unloads a specified hive file into the registry

Note The REG command is run using the permissions of the current user. If you want to use a different set of permissions, the easiest way is to log on as that user.

Querying Registry Values

Using REG query, you can read registry values by referencing the full path and name of a key or key value that you want to examine. The basic syntax is

```
reg query KeyName [/v ValueName]
```

where *KeyName* is the name of the key you want to examine and *ValueName* is an optional parameter that specifies a specific key value. In the following example, you query the DNS key under the current control set:

```
reg query HKLM\SYSTEM\CurrentControlSet\Services\DNS
```

Alternatively, if you know the specific key value you want to examine, you can limit the query results using the /V parameter. In this example, you list the value of the ImagePath entry for the DNS key:

```
reg query HKLM\SYSTEM\CurrentControlSet\Services\DNS /v ImagePath
```

The key path can also include the UNC name or IP address of a remote computer that you want to examine, such as \\Mailer1 or \\192.168.1.100. However, keep in mind that on a remote computer, you can only work with the HKLM and HKU root keys. In this example, you examine the DNS key on MAILER1:

```
reg query \\Mailer1\HKLM\SYSTEM\CurrentControlSet\Services\DNS
```

Note If you specify a nonexistent key or value, an error message is displayed. Typically, it reads: ERROR: The system was unable to find the specified registry key or value.

Comparing Registry Keys

With REG compare, you can compare registry entries and values between two systems or between two different keys on the same system. Performing registry comparisons is useful in the following situations:

- **When you are trying to troubleshoot service and application configuration issues** At such times, it is useful to compare the registry configurations between two different systems. Ideally, these systems include one that appears to be configured properly and one that you suspect is misconfigured. You can then perform a comparison of the configuration areas that you suspect are causing problems.
- **When you want to ensure that an application or service is configured the same way on multiple systems** Here you would use one system as the basis for testing the other system configurations. Ideally, the basis system is configured exactly as expected before you start comparing its configuration to other systems.

The basic syntax for REG compare is

```
reg compare KeyName1 KeyName2 [/v ValueName]
```

where *KeyName1* and *KeyName2* are the names of the subkeys that you want to compare and *ValueName* is an optional parameter that specifies a specific key value to compare. The key name can include the UNC name or IP address of a remote computer that you want to examine. In the following example, you compare the DNS key under the current control set on MAILER1 and MAILER2:

```
reg compare \\Mailer1\HKLM\SYSTEM\CurrentControlSet\Services\DNS
\\Mailer2\HKLM\SYSTEM\CurrentControlSet\Services\DNS
```

If the keys are configured the same, the output is

```
Results Compared: Identical
The operation completed successfully.
```

If the keys are configured differently, the output shows the differences. Any differences that begin with the < character pertain to the first key specified and differences that begin with the > character pertain to the second key specified. The output will also state

```
Results Compared: Different
The operation completed successfully.
```

Tip Differences are displayed because the /Od parameter is assumed by default. Using additional parameters, you can also specify that you want to see all differences and matches (/Oa), only matches (/Os), or no results (/On).

Additionally, if you want to compare all subkeys and entries recursively, you can add the /S parameter, as shown in the following example:

```
reg compare \\Mailer1\HKLM\SYSTEM\CurrentControlSet\Services\DNS
\\Mailer2\HKLM\SYSTEM\CurrentControlSet\Services\DNS /s
```

Now the key, all subkeys, and all related entries for the DNS key on MAILER1 and MAILER2 are compared.

Saving and Restoring Registry Keys

Before you modify registry entries, it is a good idea to save the keys you will use. If anything goes wrong, you can restore those keys to their original settings. To save a copy of a registry subkey and all its related subkeys and values, use REG save, as shown here:

```
reg save KeyName "FileName"
```

where *KeyName* is the path to the subkey you want to save and *FileName* is the text name of the registry hive file you want to create. The subkey path can include the UNC name or IP address of a remote computer. However, on a remote computer, you can only work with the HKLM and HKU root keys. Additionally, the filename should be enclosed in double quotation marks and should end in the .hiv extension to indicate it is a registry hive file, as shown in the following example:

```
reg save HKLM\SYSTEM\CurrentControlSet\Services\DNS "DNSKey.hiv"
```

Here, you are saving the DNS subkey and its related subkeys and values to the file named Dnskey.hiv. The filename can also include a directory path, as shown in this example:

```
reg save \HKLM\SYSTEM\CurrentControlSet\Services\DNS
"\\Mailer1\SavedData\DNSKey.hiv"
```

If the registry hive file exists, you will be prompted to overwrite the file. Press Y to overwrite. If you want to force overwrite without prompting, use the /Y parameter.

To restore a registry key that you saved previously, use Reg restore. The syntax for REG restore is

```
reg restore KeyName "FileName"
```

where *KeyName* is the path to the subkey you want to save and *FileName* is the text name of the registry hive file you want to use as the restore source. Unlike REG copy, REG restore can be used only on a local computer, meaning you cannot restore registry keys on a remote computer using the command. You can, however, start a remote

desktop session on the remote computer and then use the remote desktop logon to restore the registry key on the local computer.

An example using REG restore is shown here:

```
reg restore HKLM\SYSTEM\CurrentControlSet\Services\DNS "DNSKey.hiv"
```

Here, you are restoring the DNS key saved previously to the DNSKey.hiv file.

Adding Registry Keys

To add subkeys and values to the Windows registry, use REG add. The basic syntax for creating a key or value is

```
reg add KeyName /v ValueName /t DataType /d Data
```

where *KeyName* is the name of the key you want to examine, *ValueName* is the subkey or key value to create, *DataType* is the type of data, and *Data* is the actual value you are inserting. That seems like a lot of values, but it is fairly straightforward. Consider the following example:

```
reg add HKLM\SYSTEM\CurrentControlSet\Services\DNS /v DisplayName
/t REG_SZ /d "DNS Server"
```

Here, you add a key value called DisplayName to the DNS key in the registry. The key entry is a string with the "DNS Server" value. Note the double-quotation marks. The quotation marks are necessary in this example because the string contains a space. If the key or value you are attempting to add already exists, you are prompted to overwrite the existing data. Enter Y to overwrite, or N to cancel. To force overwriting an existing registry key or value without a prompt, use the /F parameter.

When you set expandable string values (REG_EXPAND_SZ), you must use the caret (^) to escape the percent symbols (%) that designate the environment variable you use. Consider the following example:

```
reg add HKLM\SYSTEM\CurrentControlSet\Services\DNS /v ImagePath
/t REG_EXPAND_SZ /d ^%SystemRoot^%\System32\dns.exe
```

Here, you enter **^%SystemRoot^%** so that the SystemRoot environment variable is properly entered and interpreted.

When you set non-string values, you don't need to use quotation marks, as shown in this example:

```
reg add HKLM\SYSTEM\CurrentControlSet\Services\DNS /v ErrorControl
/t REG_DWORD /d 0x00000001
```

Copying Registry Keys

Using REG copy, you can copy a registry entry to a new location on a local or remote system. The basic syntax for REG copy is

```
reg copy KeyName1 KeyName2
```

where *KeyName1* is the path to the subkey you want to copy and *KeyName2* is the path to the subkey destination. Although the subkey paths can include the UNC name or IP address of a remote computer, REG copy is limited in scope with regard to which root keys you can use when working with remote source or destination keys, as follows:

- A remote source subkey can use only the HKLM or HKU root keys.
- A remote destination subkey can use only the HKLM or HKU root keys.

In the following example, you copy the DNS subkey on the local system to the DNS subkey on MAILER2:

```
reg copy HKLM\SYSTEM\CurrentControlSet\Services\DNS
    \\Mailer2\HKLM\SYSTEM\CurrentControlSet\Services\DNS
```

By adding the /S parameter, you can copy the specified subkey as well as all subkeys and key entries under the specified subkey. In this example, the DNS subkey and all related subkey and values are copied:

```
reg copy HKLM\SYSTEM\CurrentControlSet\Services\DNS
    \\Mailer2\HKLM\SYSTEM\CurrentControlSet\Services\DNS /s
```

If values exist at the destination path, REG copy will prompt you to confirm that you want to overwrite each existing value. Press Y or N as appropriate. You can also press A to overwrite all existing values without further prompting.

> **Note** If you don't want prompts to be displayed, you can use the /F parameter to force overwrite without prompting. However, before you copy over an existing registry key, you may want to save the key so that it can be restored if problems occur. To do this, use REG save and REG restore as discussed earlier in the section of this chapter titled "Saving and Restoring Registry Keys."

Deleting Registry Keys

To delete subkeys and values from the Windows registry, use REG delete. REG delete has several different syntaxes. If you want to delete a subkey and all subkeys and entries under the subkey, use the following syntax:

```
reg delete KeyName
```

where *KeyName* is the name of the subkey you want to delete. Although the subkey path can include the UNC name or IP address of a remote computer, a remote source subkey can use only the HKLM or HKU root keys. Consider the following example:

```
reg delete \\Mailer1\HKLM\SYSTEM\CurrentControlSet\Services\DNS2
```

Here you delete the DNS2 subkey and all subkeys and entries under the subkey on MAILER1.

If you want to limit the scope of the deletion, specify that only a specific entry under the subkey should be deleted using the following syntax:

```
reg delete KeyName /v ValueName
```

where *KeyName* is the name of the subkey you want to work with and *ValueName* is the name of the specific entry to delete. As before, the subkey path can include the UNC name or IP address of a remote computer. However, a remote source subkey can use only the HKLM or HKU root keys. In this example, you delete the Description entry for the DNS2 subkey on MAILER2:

```
reg delete \\Mailer2\HKLM\SYSTEM\CurrentControlSet\Services\DNS2 /v
Description
```

> **Tip** In both cases, you will be prompted to confirm that you want to delete the specified entry permanently. Press Y to confirm the deletion. You can force deletion without prompting using the /F parameter. Another useful parameter is /Va. Using the /Va parameter, you can specify that only values under the subkey should be deleted. In this way, subkeys under the designated subkey are not deleted.

Exporting and Importing Registry Keys

Sometimes you might find it necessary or useful to copy all or part of the registry to a file and then use this copy on another computer. For example, if you've installed a component that requires extensive configuration, you might want to use it on another computer without having to go through the whole configuration process again. To do this, you would install and configure the component, export the component's registry settings from the computer, copy the settings to another computer, and then import the registry settings so that the component is properly configured. Of course, this technique works only if the complete configuration of the component is stored in the registry, but you can see how useful being able to export and import registry data can be.

When you use the REG export and REG import commands, exporting and importing registry data is fairly easy. This includes branches of data stemming from a particular root key as well as individual subkeys and the values they contain. When you export data, you create a .reg file that contains the designated registry data. This registry file is a script that can then be loaded back into the registry of this or any other computer by importing it.

Because the registry script is written as standard text, you could view it and, if necessary, modify it in any standard text editor as well. To export registry data to a file in the current directory, use the following syntax:

```
reg export KeyName FileName
```

where *KeyName* is the name of the subkey you want to work with and *FileName* is the name of the file in which to store the registry data. As before, the subkey path can include the UNC name or IP address of a remote computer. However, a remote source subkey can use only the HKLM or HKU root keys. In this example, you export the MSDTC subkey on MAILER1:

```
reg export \\Mailer1\HKLM\SOFTWARE\Microsoft\MSDTC msdtc-regkey.reg
```

You can export keys at any level of the registry. For example, you export the HKLM root key and all its subkeys using the following command line:

```
reg export HKLM hklm.reg
```

> **Tip** Add the /Y parameter to force REG export to overwrite an existing file. You can export the entire registry at the command line by typing **regedit /e *SaveFile***, where *SaveFile* is the complete file path to the location where you want to save the copy of the registry. For example, if you wanted to save a copy of the registry to C:\Save\Regdata.reg, you would type **regedit /e C:\Save\Regdata.reg**.

Importing registry data adds the contents of the registry script file to the registry of the computer you are working with, either creating new keys and values if they didn't previously exist or overwriting keys and values if they did previously exist. You can import registry data using the REG import command and the following syntax:

```
reg import FileName
```

where *FileName* is the name of the registry file in the current directory you want to import, such as:

```
reg import msdtc-regkey.reg
```

You cannot perform imports remotely or use non-local files for imports. When you are importing registry keys, you must be logged on locally to the computer and the file must exist on the local computer.

Loading and Unloading Registry Keys

Just as you sometimes must export or import registry data, you'll sometimes need to work with individual hive files. The most common reason for doing this is when you must modify a user's profile to correct an issue that prevents the user from accessing or using a system. For example, you may need to load and modify the settings for a user profile because the user inadvertently changed the display mode to an invalid setting and can no longer access the computer locally. With the user profile data loaded into the registry, you could edit the registry to correct the problem and then save the changes so that the user can once again log on to the system.

Another reason for loading registry keys is to change a particular part of the registry on a remote system. Loading and unloading hives affects only HKEY_LOCAL_MACHINE and HKEY_USERS, and you can perform these actions only when one of these root keys is selected. Rather than replacing the selected root key, the hive you are loading then becomes a subkey of that root key. HKEY_LOCAL_MACHINE and HKEY_USERS are of course used to build all the logical root keys used on a system, so you could in fact work with any area of the registry.

The file to be loaded must have been saved by the REG save command. You can load a previously saved hive file using the REG load command and the following syntax:

```
reg load RootKey\KeyName FileName
```

where *RootKey* is the root key under which the temporary key will be created, *KeyName* is the name of the temporary subkey you want to create, and *FileName* is the name of the saved hive file to load. You must create the temporary subkey under HKLM or HKU. In the following example, you create a temporary key called CurrTemp under HKLM and load the Working.hiv hive file into this key:

```
reg load HKLM\CurrTemp Working.hiv
```

You cannot perform loads remotely or use non-local files for loads. When you are loading registry keys, you must be logged on locally to the computer and the file must exist on the local computer.

Once you load a registry key, you can manipulate its subkeys and values using the techniques discussed previously. When you are finished modifying the key, you can save the key to a new registry file using REG save. After you save the key, you can unload the hive file and remove it from the computer's memory and the working registry by using the REG unload command and the following syntax:

```
reg unload RootKey\KeyName
```

where *RootKey* is the root key under which the temporary key was created and *KeyName* is the name of the temporary subkey you want to unload. In the following example, you unload the temporary key called CurrTemp under HKLM:

```
reg unload HKLM\CurrTemp
```

> **Note** You can't work with hive files that are already being used by the operating system or another process. You can, however, make a copy of the hive and then work with it. At the command line, type **reg save** followed by the abbreviated name of the root key to save and the filename to use for the hive file. For example, type **reg save hkcu c:\currhkcu.hiv** to save HKEY_LOCAL_MACHINE to a file called Currhkcu.hiv on the root folder of drive C. Although you can save the logical root keys (HKCC, HKCR, HKCU) in this manner, you can save only subkeys of HKLM and HKU for use in this technique.

Following these rules, if you needed to repair an area of the registry on a remote computer, you could:

1. Access the remote computer and save the registry hive to a file using the REG save command.
2. Copy the registry file to a folder on your computer using XCOPY or a similar command.
3. Load the related hive file into the registry of your computer using the REG load command.
4. Make any necessary changes and then save the changes using the REG save command.

5. Import the registry hive on the remote computer to repair the problem using the REG import command.
6. After you test the changes on the remote computer, unload the registry hive from your computer using the REG unload command.

Managing System Services

Services provide key functions to workstations and servers. To manage system services on local and remote systems, you'll use the service controller command SC, which has several subcommands, only some of which are explored here. The sections that follow discuss each of these subcommands:

- **SC config** Configures service startup and logon accounts
- **SC query** Displays the list of all services configured on the computer
- **SC qc** Displays the configuration of a specific service
- **SC start** Starts services
- **SC stop** Stops services
- **SC pause** Pauses services
- **SC continue** Resumes services
- **SC failure** Sets the actions to take upon failure of a service
- **SC qfailure** Views the actions to take upon failure of a service

With all commands, you can specify the name of the remote computer whose services you want to work with. To do this, insert the UNC name or IP address of the computer before the subcommand you want to use. This makes the syntax

```
sc ServerName Subcommand
```

Viewing Configured Services

To get a list of all services configured on a system, type the following command at the command prompt:

```
sc query type= service state= all
```

or

```
sc ServerName query type= service state= all
```

where *ServerName* is the UNC name or IP address of the remote computer, such as \\Mailer1 or \\192.168.1.100, as shown in the following examples:

```
sc \\Mailer1 query type= service state= all
sc \\192.168.1.100 query type= service state= all
```

Note You must include a space after the equal sign (=) as used with type= *service* and *state= all*. If you don't use a space, the command will fail.

With the *state* flag, you can also use the value *active* (to show running services only) or *inactive* (to show all paused or stopped services). Consider the following examples:

```
sc \\Mailer1 query type= service state= active
sc \\Mailer1 query type= service state= inactive
```

In the first example, you query MAILER1 for a list of all services that are running. In the second example, you query MAILER1 for a list of all services that are stopped.

The output of SC query shows the services and their configurations. Each service entry is formatted as follows:

```
SERVICE_NAME: W3SVC
DISPLAY_NAME: World Wide Web Publishing Service
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```

As an administrator, the fields you will work with the most are

- **Service Name** The abbreviated name of the service. Only services installed on the system are listed here. If a service you need isn't listed, you'll need to install it.
- **Display Name** The descriptive name of the service.
- **State** The state of the service as Running, Paused, or Stopped.

As you'll see if you run the SC query command, the output is very long and is best used with a filter to get only the information you want to see. For example, if you use the following command, you clean up the output to show only the most important fields:

```
sc query type= service | find /v "x0"
```

Here you pipe the output of SC query through the FIND command and clean up the output so the service entries appear, as shown in this example:

```
SERVICE_NAME: W3SVC
DISPLAY_NAME: World Wide Web Publishing Service
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, PAUSABLE, ACCEPTS_SHUTDOWN)
```

Note The parameter /V "x0" tells the FIND command to display only lines of output that do not contain the text x0, which is the common text on the WIN32_Exit_Code, Service_Exit_Code, Checkpoint, and Wait_Hint fields. By specifying that you don't want to see lines of output that contain this value, you therefore remove these unwanted fields from the display.

If you know the name of a service you want to work with, you can use SC qc to display its configuration information. The syntax is

```
sc qc ServiceName
```

where *ServiceName* is the name of the service you want to examine. The output for individual services looks like this:

```
SERVICE_NAME: w3svc
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\WINDOWS\System32\svchost.exe -k iissvcs
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : World Wide Web Publishing Service
        DEPENDENCIES       : RPCSS
                           : HTTPFilter
                           : IISADMIN
        SERVICE_START_NAME : LocalSystem
```

Note that the output doesn't tell you the current status of the service. It does, however, tell you the following:

- **Binary Path Name** The file path to the executable for the service
- **Dependencies** Services that cannot run unless the specified service is running
- **Display Name** The descriptive name of the service
- **Service Start Name** The name of the user account the service logs on as
- **Start Type** The startup configuration of the service

  Note Services that are configured to start automatically are listed as AUTO_START. Services that are configured to start manually are listed as DEMAND_START. Services that are disabled are listed as DISABLED.

- **Type** The type of service and whether it is a shared process

Note When you are configuring a service logon, it is sometimes important to know whether a process runs in its own context or is shared. Shared processes are listed as WIN32_SHARE_PROCESS. Processes that run in their own context are listed as WIN32_OWN_PROCESS.

Starting, Stopping, and Pausing Services

As an administrator, you'll often have to start, stop, or pause Windows services. The related SC commands and their syntaxes are

Start a service:

```
sc start ServiceName
```

Pause a service:

```
sc pause ServiceName
```

Resume a paused service:

```
sc continue ServiceName
```

Stop a service:

```
sc stop ServiceName
```

where *ServiceName* in each case is the abbreviated name of the service you want to work with, such as

```
sc start w3svc
```

As with all SC commands, you can also specify the name of the remote computer whose services you want to work with. For example, to start the w3svc on MAILER1, you would use the following command:

```
sc \\Mailer1 start w3svc
```

The state listed in the results should show START_PENDING. With stop, pause, and continue you'll see STOP_PENDING, PAUSE_PENDING, and CONTINUE_PENDING respectively as well. If an error results, the output states FAILED and error text is provided to describe the reason for the failure in more detail. If you are trying to start a service that is already started, you'll see the error

```
An instance of the service is already running.
```

If you are trying to stop a service that is already stopped, you'll see the error

```
The service has not been started.
```

Configuring Service Startup

You can set Windows services to start manually or automatically. You can also turn them off permanently by disabling them. You configure service startup using

```
sc config ServiceName start= flag
```

where *ServiceName* is the abbreviated name of the service you want to work with and flag is the startup type to use. For services, valid flag values are

- **Auto** Starts service at system startup
- **Demand** Allows the services to be started manually
- **Disabled** Turns off the service
- **Delayed-Auto** Delays the start of the service until all non-delayed automatic services have started

Following this, you can configure a service to start automatically by using

```
sc config w3svc start= auto
```

or

```
sc \\Mailer1 config w3svc start= auto
```

> **Note** You must include a space after the equal sign (=) as used with *start= auto*. If you don't use a space, the command will fail. Note also the command only reports SUCCESS or FAILURE. It won't tell you that the service was already configured in the startup mode you've specified.

Disabling a service doesn't stop a running service. It only prevents it from being started the next time the computer is booted. To ensure that the service is disabled and stopped, run SC stop and then SC config.

Configuring Service Logon

You can configure Windows services to log on as a system account or as a specific user. To ensure a service logs on as the LocalSystem account, use

```
sc config ServiceName obj= LocalSystem
```

where *ServiceName* is the name of the service you are configuring to use the LocalSystem account. If the service provides a user interface that can be manipulated, add the flags **type= interact type= own**, as shown in the following example:

```
sc config w3svc obj= LocalSystem type= interact type= own
```

The *type= interact* flag specifies that the service is allowed to interact with the Windows desktop. The *type= own* flag specifies that the service runs in its own process. In the case of a service that shares its executable files with other services, you would use the *type= share* flag, as shown in this example:

```
sc config w3svc obj= LocalSystem type= interact type= share
```

> **Tip** If you don't know whether a service runs as a shared process or in its own context, use SC qc to determine the service's start type. This command is discussed in the section titled "Viewing Configured Services," earlier in this chapter.

Services can also log on using named accounts. To do this, use

```
sc config ServiceName obj= [Domain\]User password= Password
```

where *Domain* is the optional domain name in which the user account is located, *User* is the name of the user account whose permissions you want to use, and *Password* is the password of that account. Consider the following example:

```
sc config w3svc obj= adatum\webbies password= blue5!CraZy
```

Here, you configure W3svc to use the Webbies account in the Adatum domain. The output of the command should state SUCCESS or FAILED. The change will fail if the account name is invalid or doesn't exist, or if the password for the account is invalid.

> **Note** If a service has been previously configured to interact with the desktop under the LocalSystem account, you cannot change the service to run under a domain account without using the type= own flag. The syntax therefore becomes sc config **ServiceName** obj= [**Domain**\]**User** password= Password type= own.

> **Real World** As an administrator, you should keep track of any accounts that are used with services. These accounts can be the source of huge security problems if they're not configured properly. Service accounts should have the strictest security settings and as few permissions as possible while allowing the service to perform necessary functions. Typically, accounts used with services don't need many of the permissions you would assign to a normal user account. For example, most service accounts don't need the right to log on locally. Every administrator should know what service accounts are used (so that he or she can better track use of these accounts), and the accounts should be treated as if they were administrator accounts. This means secure passwords, careful monitoring of account usage, careful application of account permissions and privileges, and so on.

Configuring Service Recovery

Using the SC failure command, you can configure Windows services to take specific actions when a service fails. For example, you can attempt to restart the service or run an application.

You can configure recovery options for the first, second, and subsequent recovery attempts. The current failure count is incremented each time a failure occurs. You can also set a parameter that specifies the time that must elapse before the failure counter is reset. For example, you could specify that if 24 hours have passed since the last failure, the failure counter should be reset.

Before you try to configure service recovery, check the current recovery settings using SC qfailure. The syntax is

```
sc qfailure ServiceName
```

where *ServiceName* is the name of the service you want to work with, such as

```
sc qfailure w3svc
```

You can of course specify a remote computer as well, such as

```
sc \\Mailer1 qfailure w3svc
```

or

```
sc \\192.168.1.100 qfailure w3svc
```

In the output, the failure actions are listed in the order they are performed. In the following example output, W3svc is configured to attempt to restart the service the first and second time the service fails and to restart the computer if the service fails a third time:

```
[SC] QueryServiceConfig2 SUCCESS

SERVICE_NAME: w3svc
       RESET_PERIOD (in seconds)    : 86400
       REBOOT_MESSAGE               :
       COMMAND_LINE                 :
        FAILURE_ACTIONS               : RESTART -- Delay = 1 milliseconds.
                                        RESTART -- Delay = 1 milliseconds.
                                        REBOOT -- Delay = 1000 milliseconds.
```

Note Windows automatically configures recovery for some critical system services during installation. Typically, these services are configured so that they attempt to restart the service. A few services are configured so that they run programs. For example, the IIS Admin service is configured to run a program called Iisreset.exe if the service fails. This program is an application that corrects service problems and safely manages dependent IIS services while working to restart the IIS Admin service.

The command you use to configure service recovery is SC failure and its basic syntax is

```
sc failure ServiceName reset= FailureResetPeriod actions= RecoveryActions
```

where *ServiceName* is the name of the service you are configuring, *FailureResetPeriod* specifies the time, in seconds, that must elapse without failure in order to reset the failure counter, and *RecoveryActions* are the actions to take when failure occurs plus the delay time (in milliseconds) before that action is initiated. The available recovery actions are

- **Take No Action (indicated by an empty string "")** The operating system won't attempt recovery for this failure but might still attempt recovery of previous or subsequent failures.
- **Restart The Service** Stops and then starts the service after a brief pause.
- **Run A Program** Allows you to run a program or a script in case of failure. The script can be a batch program or a Windows script. If you select this option, set the full file path to the program you want to run and then set any necessary command-line parameters to pass in to the program when it starts.

- **Reboot The Computer** Shuts down and then restarts the computer after the specified delay time is elapsed.

Best Practices When you configure recovery options for critical services, you might want to try to restart the service on the first and second attempts and then reboot the server on the third attempt.

When you work with SC failure, keep the following in mind:

- **The reset period is set in seconds.** Reset periods are commonly set in multiples of hours or days. An hour is 3,600 seconds and a day is 86,400 seconds. For a two-hour reset period, for example, you'd use the value 7,200.
- **Each recovery action must be followed by the time to wait (in milliseconds) before performing the action.** For a service restart you'll probably want to use a short delay, such as 1 millisecond (no delay), 1 second (1,000 milliseconds), or 5 seconds (5,000 milliseconds). For a restart of the computer, you'll probably want to use a longer delay, such as 15 seconds (15,000 milliseconds) or 30 seconds (30,000 milliseconds).
- **Enter the actions and their delay times as a single text entry with each value separated by a forward slash (/).** For example, you could use the value: restart/1000/restart/1000/ reboot/15000. Here, on the first and second attempts the service is restarted after a 1-second delay, and on the third attempt the computer is rebooted after a 15-second delay.

Consider the following examples:

```
sc failure w3svc reset= 86400 actions= restart/1/restart/1/reboot/30000
```

Here, on the first and second attempts the service is restarted almost immediately, and on the third attempt the computer is rebooted after a 30-second delay. In addition, the failure counter is reset if no failures occur in a 24-hour period (86,400 seconds). You can also specify a remote computer by inserting the UNC name or IP address as shown in previous examples.

If you use the Run action, you specify the command or program to run using the *Command*= parameter. Follow the *Command*= parameter with the full file path to the command to run and any arguments to pass to the command. Be sure to enclose the command path and text in double quotation marks, as in the following example:

```
sc failure w3svc reset= 86400 actions= restart/1/restart/1/run/30000
command= "c:\restart_w3svc.exe 15"
```

Restarting and Shutting Down Systems from the Command Line

You'll often find that you need to shut down or restart systems. One way to do this is to use the Shutdown utility, which you can use to work with both local and remote systems. Another way to manage system shutdown or restart is to schedule a shutdown. Here, you can use Schtasks to specify when shutdown should be run or you can create a script with a list of shutdown commands for individual systems.

Real World Although Windows systems usually start up and shut down without problems, they can occasionally stop responding during these processes. If this happens, try to determine the cause. Some of the reasons systems might stop responding include the following:

1. The system is attempting to execute or is running a startup or shutdown script that has not completed or is itself not responding (and in this case, the system might be waiting for the script to time out).
2. A startup initialization file or service may be the cause of the problem and if so, you might need to troubleshoot startup items using the System Configuration Utility (Msconfig). Disabling a service, startup item, or entry in a startup initialization file might also solve the problem.
3. The system may have an antivirus program that is causing the problem. In some cases, the antivirus program may try to scan the removable media drives when you try to shut down the system. To resolve this, configure the antivirus software so that it doesn't scan the removable media drives or other drives with removable media on shutdown. You could also try temporarily disabling or turning off the antivirus program.
4. Improperly configured sound devices can cause startup and shutdown problems. To determine what the possible source is, examine each of these devices in turn. Turn off sound devices and then restart the computer. If the problem clears up, you have to install new drivers for the sound devices you are using or you may have a corrupted Start Windows or Exit Windows sound file.
5. Improperly configured network cards can cause startup and shutdown problems. Try turning off the network adapter and restarting. If that works, you might need to remove and then reinstall the adapter's driver or obtain a new driver from the manufacturer.
6. Improperly configured video adapter drivers can cause startup and shutdown problems. From another computer, remotely log on and try to roll back the current video drivers to a previous version. If that's not possible, try uninstalling and then reinstalling the video drivers.

Managing Restart and Shutdown of Local Systems

On a local system, you can manage shutdown and restart using the following commands:

Shutdown local system:

```
shutdown /s /t ShutdownDelay /l /f
```

Restart local system:

```
shutdown /r /t ShutdownDelay /l /f
```

Cancel delayed shutdown of local computer:

```
shutdown /a
```

where /T *ShutdownDelay* is used to set the optional number of seconds to wait before shutdown or restart, /L optionally logs off the current user immediately, and /F optionally forces running applications to close without warning users in advance. In this example, the local system is restarted after a 60-second delay:

```
shutdown /r /t 60
```

> **Best Practices** In most network environments, system uptime is of the utmost importance. Systems that are restarting or shutting down aren't available to users, which might mean someone won't be able to finish her work and might get upset as a result. Rather than shut down systems in the middle of business hours, consider performing shutdowns before or after normal business hours. But if you need to shut down a system during business hours, warn users beforehand if possible, allowing them to save current work and log off the system as necessary.

Managing Restart and Shutdown of Remote Systems

With remote systems, you need to specify the UNC name or IP address of the system you want to shut down or restart using the /M parameter. Thus, the basic syntax for shutdown, restart, and cancel delayed shutdown needs to be modified as shown in these examples:

Shutdown remote system:

```
shutdown /s /t ShutdownDelay /l /f /m \\System
```

Restart remote system:

```
shutdown /r /t ShutdownDelay /l /f /m \\System
```

Cancel delayed shutdown of remote computer:

```
shutdown /a /m \\System
```

In this example, MAILER1 is restarted after a 30-second delay:

```
shutdown /r /t 30 /m \\Mailer1
```

In this example, the system with the IP address 192.168.1.105 is restarted immediately and running applications are forced to stop running:

```
shutdown /r /f /m \\192.168.1.105
```

Adding Shutdown or Restart Reasons and Comments

In most network environments, it's a good idea to document the reasons for shutting down or restarting computers. Following an unplanned shutdown, you can document the shutdown in the computer's system log by expanding the syntax to include the following parameters:

```
/e /c "UnplannedReason" /d MajorCode:MinorCode
```

where /E replaces the /R switch, */C "UnplannedReason"* sets the detailed reason (which can be up to 512 characters in length) for the shutdown or restart, and */D MajorCode:MinorCode* sets the reason code for the shutdown. Reason codes are arbitrary, with valid major codes ranging from 0 to 255 and valid minor reason codes ranging from 0 to 65,535.

For a planned restart, consider the following example:

```
shutdown /r /m \\Mailer1 /c "System Reset" /d 5:15
```

In this example, you are restarting MAILER1 and documenting the reason for the unplanned restart as a "System Reset" using the reason code 5:15.

Table 5-3 summarizes the common reasons and codes for shutdowns and restarts for Windows Vista and Windows Server 2008. As the table shows, Windows can generate the prefix code E for Expected, U for Unexpected, and P for planned as well as various combinations of these prefix codes.

Table 5-3 Common Reasons and Codes for Shutdowns and Restarts

Prefix Code	Major Code	Minor Code	Shutdown or Restart Type
U	0	0	Other (Unplanned)
E	0	0	Other (Unplanned)
E P	0	0	Other (Planned)
U	0	5	Other Failure: System Unresponsive
E	1	1	Hardware: Maintenance (Unplanned)
E P	1	1	Hardware: Maintenance (Planned)
E	1	2	Hardware: Installation (Unplanned)
E P	1	2	Hardware: Installation (Planned)
P	2	3	Operating System: Upgrade (Planned)
E	2	4	Operating System: Reconfiguration (Unplanned)
E P	2	4	Operating System: Reconfiguration (Planned)
P	2	16	Operating System: Service pack (Planned)
U	2	17	Operating System: Hotfix (Unplanned)
P	2	17	Operating System: Hotfix (Planned)
U	2	18	Operating System: Security fix (Unplanned)
P	2	18	Operating System: Security fix (Planned)
E	4	1	Application: Maintenance (Unplanned)
E P	4	1	Application: Maintenance (Planned)
E P	4	2	Application: Installation (Planned)
E	4	5	Application: Unresponsive
E	4	6	Application: Unstable

Table 5-3 Common Reasons and Codes for Shutdowns and Restarts

Prefix Code	Major Code	Minor Code	Shutdown or Restart Type
U	5	15	System Failure: Stop error
E	5	19	Security issue
U	5	19	Security issue
E P	5	19	Security issue (Planned)
E	5	20	Loss of network connectivity (Unplanned)
U	6	11	Power Failure: Cord Unplugged
U	6	12	Power Failure: Environment
P	7	0	Legacy API shutdown (Planned)

For the SHUTDOWN command, only the P: and the U: prefixes are accepted. For example, with planned shutdowns and restarts, prefix the reason codes with **p:** to indicate a planned shutdown, as shown here:

```
/c "PlannedReason" /d p:MajorCode:MinorCode
```

For instance, consider the following code:

```
shutdown /r /m \\Mailer1 /c "Planned Application Upgrade" /d p:4:2
```

In this example, you are restarting MAILER1 and documenting the reason for the planned restart as a "Planned Application Upgrade" using the reason code 4:2.

Chapter 7

Monitoring Processes and Maintaining Performance

An important part of every administrator's job is to monitor network systems and ensure that everything is running smoothly—or as smoothly as can be expected, anyway. As you learned in the previous chapter, watching the event logs closely can help you detect and track problems with applications, security, and essential services. Often when you detect or suspect a problem, you'll need to dig deeper to search out the cause of the problem and correct it. Hopefully, by pinpointing the cause of a problem, you can prevent it from happening again.

Managing Applications, Processes, and Performance

Whenever the operating system or a user starts a service, runs an application, or executes a command, Windows starts one or more processes to handle the related program. Several command-line utilities are available to help you manage and monitor programs. These utilities include the following:

- **Task List (Tasklist)** Lists all running processes by name and process ID. Includes information on the user session and memory usage.
- **Task Kill (Taskkill)** Stops running processes by name or process ID. Using filters, you can also halt processes by process status, session number, CPU time, memory usage, user name, and more.
- **Powershell get-process** Displays performance statistics, including memory and CPU usage, as well as a list of all running processes. Used to get a detailed snapshot of resource usage and running processes. Available when you install Windows PowerShell.

In the sections that follow, you'll find detailed discussions on how these command-line tools are used. First, however, let's look at the ways processes are run and the common problems you may encounter when working with them.

Understanding System and User Processes

Generally, processes that the operating system starts are referred to as *system processes*; processes that users start are referred to as *user processes*. Most user processes are run in interactive mode. That is, a user starts a process interactively with the keyboard or mouse. If the application or program is active and selected, the related interactive process has control over the keyboard and mouse until you switch control by terminating the

program or selecting a different one. When a process has control, it's said to be running "in the foreground."

Processes can also run in the background, independently of user logon sessions. Background processes do not have control over the keyboard, mouse, or other input devices and are usually run by the operating system. Using the Task Scheduler, users can run processes in the background as well, however, and these processes can operate regardless of whether the user is logged on. For example, if Task Scheduler starts a scheduled task while the user is logged on, the process can continue even when the user logs off.

Windows tracks every process running on a system by image name, process ID, priority, and other parameters that record resource usage. The image name is the name of the executable that started the process, such as Msdtc.exe or Svchost.exe. The process ID is a numeric identifier for the process, such as 2588. The process ID is a priority-indicator of how much of the system's resources the process should get relative to other running processes. With priority processing, a process with a higher priority gets preference over processes with lower priority and may not have to wait to get processing time, access memory, or work with the file system. A process with lower priority, on the other hand, usually must wait for a higher-priority process to complete its current task before gaining access to the CPU, memory, or the file system.

In a perfect world, processes would run perfectly and would never have problems. The reality is, however, that problems occur and they often appear when you'd least want them to. Common problems include the following:

- Processes become nonresponsive, such as when an application stops processing requests. When this happens, users may tell you that they can't access a particular application, that their requests aren't being handled, or that they were kicked out of the application.
- Processes fail to release the CPU, such as when you have a runaway process that is using up CPU time. When this happens, the system may appear to be slow or nonresponsive because the runaway process is hogging processor time and is not allowing other processes to complete their tasks.
- Processes use more memory than they should, such as when an application has a memory leak. When this happens, processes aren't properly releasing memory that they're using. As a result, the system's available memory may gradually decrease over time and as the available memory gets low, the system may be slow to respond to requests or it may become nonresponsive. Memory leaks can also make other programs running on the same system behave erratically.

In most cases, when you detect these or other problems with system processes, you'll want to stop the process and start it again. You would also want to examine the event logs to see whether you can determine the cause of the problem. In the case of memory leaks, you want to report the memory leak to the developers and see whether an update that resolves the problem is available.

A periodic restart of an application with a known memory leak is often useful. Restarting the application should allow the operating system to recover any lost memory.

Examining Running Processes

When you want to examine processes that are running on a local or remote system, you can use the Tasklist command-line utility. With Tasklist, you can do the following:

Obtain the process ID, status, and other important information about processes running on a system.

- View the relationship between running processes and services configured on a system.
- View lists of DLLs used by processes running on a system.
- Use filters to include or exclude processes from Tasklist queries.

Each of these tasks is discussed in the sections that follow.

Obtaining Detailed Information on Processes

On a local system, you can view a list of running tasks simply by typing **tasklist** at the command prompt. As with many other command-line utilities, Tasklist runs by default with the permissions of the currently logged-on user, and you can also specify the remote computer whose tasks you want to query and the Run As permissions. To do this, use the expanded syntax, which includes the following parameters:

/s *Computer* /u [*Domain*\]*User* [/p *Password*]

where *Computer* is the remote computer name or IP address, *Domain* is the optional domain name in which the user account is located, *User* is the name of the user account whose permissions you want to use, and *Password* is the optional password for the user account. If you don't specify the domain, the current domain is assumed. If you don't provide the account password, you are prompted for the password.

To see how you can add computer and user information to the syntax, consider the following examples:

Query Mailer1 for running tasks:

```
tasklist /s mailer1
```

Query 192.168.1.5 for running tasks using the account adatum\wrstanek:

```
tasklist /s 192.168.1.5 /u adatum\wrstanek
```

The basic output of these commands is in table format. You can also format the output as a list or lines of comma-separated values using /Fo List or /Fo Csv, respectively. Remember that you can redirect the output to a file using output redirection (> or >>), such as tasklist /s mailer1 >> current-tasks.log.

Regardless of whether you are working with a local or remote computer, the output should be similar to the following:

```
Image Name                     PID Session Name        Session#    Mem Usage
========================= ======== ================ =========== ============
System Idle Process              0 Services                   0         28 K
System                           4 Services                   0     28,952 K
smss.exe                       488 Services                   0        776 K
csrss.exe                      560 Services                   0      5,272 K
wininit.exe                    608 Services                   0      4,056 K
csrss.exe                      620 Console                    1     13,004 K
services.exe                   652 Services                   0      7,456 K
lsass.exe                      664 Services                   0      1,852 K
lsm.exe                        680 Services                   0      6,400 K
svchost.exe                    836 Services                   0      7,228 K
winlogon.exe                   868 Console                    1      5,544 K
svchost.exe                    932 Services                   0      9,440 K
svchost.exe                    984 Services                   0     23,304 K
svchost.exe                   1048 Services                   0     12,208 K
svchost.exe                   1100 Services                   0     71,696 K
svchost.exe                   1132 Services                   0     36,920 K
dwm.exe                       2832 Console                    1     65,456 K
explorer.exe                  2892 Console                    1     25,624 K
```

The Tasklist fields provide the following information:

- **Image Name** The name of the process or executable running the process.
- **PID** The process identification number.
- **Session Name** The name of the session from which the process is being run. An entry of *console* means the process was started locally.
- **Session #** A numerical identifier for the session.
- **Memory Usage** The total amount of memory being used by the process at the specific moment that Tasklist was run.

If you want more detailed information you can specify that verbose mode should be used by including the /V parameter. Verbose mode adds the following columns of data:

- **Status** Current status of the process as Running, Not Responding, or Unknown. A process can be in an Unknown state and still be running and responding normally. A process that is Not Responding, however, more than likely must be stopped or restarted.

- **User Name** User account under which the process is running, listed in domain\user format. For processes started by Windows, you will see the name of the system account used, such as SYSTEM, LOCAL SERVICE, or NETWORK SERVICE, with the domain listed as NT AUTHORITY.
- **CPU Time** The total amount of CPU-cycle time used by the process since its start.
- **Window Title** Windows display name of the process if available. Otherwise, the display name is listed as *N/A* for not available. For example, the HelpPane.exe process is listed with the title Windows Help And Support Center

If you use Tasklist to examine running processes, you'll note two unique processes: System and System Idle Process. System shows the resource usage for the local system process. System Idle Process tracks the amount of CPU processing time that isn't being used. Thus, a 99 in the CPU column for the System Idle Process means 99 percent of the system resources currently aren't being used. If you believe that a system is overloaded, you should monitor the idle process. Watch the CPU usage and the total CPU time. If the system consistently has low idle time (meaning high CPU usage), you may want to consider upgrading the processor or even adding processors.

As you examine processes, keep in mind that a single application might start multiple processes. Generally, these processes are dependent on a central process, and from this main process a process tree containing dependent processes is formed. When you terminate processes, you'll usually want to target the main application process or the application itself rather than dependent processes. This ensures that the application is stopped cleanly.

Viewing the Relationship Between Running Processes and Services

When you use Tasklist with the /Svc parameter, you can examine the relationship between running processes and services configured on the system. In the output, you'll see the process image name, process ID, and a list of all services that are using the process, similar to that shown in the following example:

```
Image Name                     PID Services
========================= ======= ====================================
System Idle Process              0 N/A
System                           4 N/A
smss.exe                       488 N/A
csrss.exe                      560 N/A
wininit.exe                    608 N/A
csrss.exe                      620 N/A
services.exe                   652 N/A
lsass.exe                      664 KeyIso, ProtectedStorage, SamSs
lsm.exe                        680 N/A
svchost.exe                    836 DcomLaunch, PlugPlay
winlogon.exe                   868 N/A
```

```
svchost.exe                  932 RpcSs
svchost.exe                  984 WinDefend
svchost.exe                 1048 Audiosrv, Dhcp, Eventlog, lmhosts, wscsvc
svchost.exe                 1100 AudioEndpointBuilder, CscService, EMDMgmt,
                                 Netman, PcaSvc, SysMain,
                                 TabletInputService, TrkWks, UmRdpService,
                                 UxSms, WdiSystemHost, Wlansvc, WPDBusEnum,
                                 wudfsvc
svchost.exe                 1132 AeLookupSvc, BITS, Browser, CertPropSvc,
                                 EapHost, gpsvc, IKEEXT, iphlpsvc,
                                 LanmanServer, MMCSS, ProfSvc, RasMan,
                                 Schedule, seclogon, SENS, SessionEnv,
                                 ShellHWDetection, Themes, Winmgmt, wuauserv
svchost.exe                 1384 EventSystem, fdPHost, FDResPub,
                                 LanmanWorkstation, netprofm, nsi, SSDPSRV,
                                 upnphost, W32Time, WebClient
svchost.exe                 1520 CryptSvc, Dnscache, KtmRm, NlaSvc, TapiSrv,
                                 TermService
spoolsv.exe                 1776 Spooler
svchost.exe                 1800 BFE, DPS, MpsSvc
dwm.exe                     2832 N/A
explorer.exe                2892 N/A
```

By default, the output is formatted as a table, and you cannot use the *list* or *CSV* format. Beyond formatting, the important thing to note here is that services are listed by their abbreviated names, which is the naming style used by Sc, the service controller command-line utility, to manage services.

You can use the correlation between processes and services to help you manage systems. For example, if you think you are having problems with the World Wide Web Publishing Service (W3svc), one step in your troubleshooting process is to begin monitoring the service's related process or processes. You would want to examine the following:

- Process status
- Memory usage
- CPU time

By tracking these statistics over time, you can watch for changes that could indicate the process has stopped responding or is a runaway process hogging CPU time, or that there is a memory leak.

Viewing Lists of DLLs Being Used by Processes

When you use Tasklist with the /M parameter, you can examine the relationship between running processes and DLLs configured on the system. In the output, you'll see the process image name, process ID, and a list of all DLLs that the process is using, as shown in the following example:

```
Image Name                     PID Modules
========================= ======= ========================================
System Idle Process              0 N/A
System                           4 N/A
smss.exe                       488 N/A
csrss.exe                      560 N/A
wininit.exe                    608 N/A
csrss.exe                      620 N/A
services.exe                   652 N/A
lsass.exe                      664 N/A
lsm.exe                        680 N/A
svchost.exe                    836 N/A
winlogon.exe                   868 N/A
svchost.exe                    932 N/A
svchost.exe                    984 N/A
svchost.exe                   1048 N/A
svchost.exe                   1100 N/A
svchost.exe                   1132 N/A
dwm.exe                       2832 ntdll.dll, kernel32.dll, ADVAPI32.dll,
                                   RPCRT4.dll, GDI32.dll, USER32.dll,
                                   msvcrt.dll, ole32.dll, OLEAUT32.dll,
                                   UxTheme.dll, IMM32.dll, MSCTF.dll,
                                   dwmredir.dll, SLWGA.dll, urlmon.dll,
                                   SHLWAPI.dll, iertutil.dll, WTSAPI32.dll,
                                   slc.dll, LPK.DLL, USP10.dll, comctl32.dll,
                                   NTMARTA.DLL, WLDAP32.dll, WS2_32.dll,
                                   NSI.dll, PSAPI.DLL, SAMLIB.dll,
                                   milcore.dll, dwmapi.dll, uDWM.dll,
                                   WindowsCodecs.dll, ctagent.dll, d3d9.dll,
                                   VERSION.dll, d3d8thk.dll, nvd3dum.dll,
                                   IconCodecService.dll
explorer.exe                  2892 ntdll.dll, kernel32.dll, ADVAPI32.dll,
                                   RPCRT4.dll, GDI32.dll, USER32.dll,
                                   msvcrt.dll, SHLWAPI.dll, SHELL32.dll,
                                   ole32.dll, OLEAUT32.dll, SHDOCVW.dll,
                                   UxTheme.dll, POWRPROF.dll, dwmapi.dll,
                                   gdiplus.dll, slc.dll, PROPSYS.dll,
                                   BROWSEUI.dll, IMM32.dll, MSCTF.dll,
                                   DUser.dll, LPK.DLL, USP10.dll,
                                   comctl32.dll, WindowsCodecs.dll,
                                   apphelp.dll, CLBCatQ.DLL, cscui.dll,
                                   CSCDLL.dll, CSCAPI.dll,
                                   IconCodecService.dll, Secur32.dll,
                                   rsaenh.dll, msiltcfg.dll, VERSION.dll,
                                   msi.dll, NTMARTA.DLL, WLDAP32.dll,
                                   WS2_32.dll, NSI.dll, PSAPI.DLL, SAMLIB.dll,
                                   SFC.DLL, sfc_os.dll, SETUPAPI.dll,
                                   timedate.cpl, ATL.DLL, NETAPI32.dll,
                                   OLEACC.dll, actxprxy.dll, USERENV.dll
```

Knowing which DLL modules a process has loaded can further help you pinpoint what may be causing a process to become nonresponsive, to fail to release the CPU, or to use more memory than it should. In some cases, you might want to check DLL versions to ensure that they are the correct DLLs that the system should be running. Here, you would need to consult the Microsoft Knowledge Base or manufacturer documentation to verify DLL versions and other information.

If you are looking for processes using a specified DLL, you can also specify the name of the DLL you are looking for. For example, if you suspect that the printer spooler driver Winspool.drv is causing processes to hang up, you can search for processes that use Winspool.drv instead of Winspool32.drv and check their status and resource usage.

The syntax that you use to specify the DLL to find is

```
tasklist /m DLLName
```

where *DLLName* is the name of the DLL to search for. Tasklist matches the DLL name without regard to the letter case, and you can enter the DLL name in any letter case. Consider the following example:

```
tasklist /m winspool.drv
```

In this example, you are looking for processes using Winspool.drv. The output of the command would show the processes using the DLL along with their process IDs, as shown in the following example:

```
Image Name                     PID Modules
========================= ======== ====================================
explorer.exe                  2892 WINSPOOL.DRV
rundll32.exe                  3308 WINSPOOL.DRV
acrotray.exe                  3340 WINSPOOL.DRV
IAAnotif.exe                  3464 WINSPOOL.DRV
IntelHCTAgent.exe             3584 WINSPOOL.DRV
DrgToDsc.exe                  3636 WINSPOOL.DRV
WINWORD.EXE                   4836 WINSPOOL.DRV
```

Filtering Task List Output

Using the /Fi parameter of the Tasklist utility, you can filter task lists using any of the information fields available, even if the information field isn't normally included in the output because of the parameters you've specified. This means you can specify that you want to see only processes listed with a status of Not Responding, only information for Svchost.exe processes, or only processes that use a large amount of CPU time.

You designate how a filter should be applied to a particular Tasklist information field using filter operators. The following filter operators are available:

- **Eq** Equals. If the field contains the specified value, the process is included in the output.
- **Ne** Not equals. If the field contains the specified value, the process is excluded from the output.

- **Gt** Greater than. If the field contains a numeric value and that value is greater than the value specified, the process is included in the output.
- **Lt** Less than. If the field contains a numeric value and that value is less than the value specified, the process is included in the output.
- **Ge** Greater than or equal to. If the field contains a numeric value and that value is greater than or equal to the value specified, the process is included in the output.
- **Le** Less than or equal to. If the field contains a numeric value and that value is less than or equal to the value specified, the process is included in the output.

As Table 7-1 shows, the values that you can use with filter operators depend on the task list information field you use. Remember that all fields are available even if they aren't normally displayed with the parameters you've specified. For example, you can match the status field without using the */V* (verbose) flag.

Table 7-1 Filter Operators and Valid Values for Tasklist

Filter Field Name	Valid Operators	Valid Values
CPUTime	eq, ne, gt, lt, ge, le	Any valid time in the format hh:mm:ss
Services	eq, ne	Any valid string of characters
ImageName	eq, ne	Any valid string of characters
MemUsage	eq, ne, gt, lt, ge, le	Any valid integer, expressed in kilobytes (KB)
Modules	eq, ne	DLL name
PID	eq, ne, gt, lt, ge, le	Any valid positive integer
Session	eq, ne, gt, lt, ge, le	Any valid session number
SessionName	eq, ne	Any valid string of characters
Status	eq, ne	Running, Not Responding, Unknown
Username	eq, ne	Any valid user name, with user name only or in domain\user format
WindowTitle	eq, ne	Any valid string of characters

You must use double quotation marks to enclose the filter string. Consider the following examples to see how you can use filters:

Look for processes that are not responding:

```
tasklist /fi "status eq not responding"
```

When working with remote systems, you can't filter processes by status or window title. A workaround for this in some cases is to pipe the output through the FIND command, such as **tasklist /v /s Mailer1 /u adatum\wrstanek | find /i "not**

responding". Note that in this case, the field you are filtering must be in the output, which is why the /V parameter was added to the example. Further, you should specify that the FIND command should ignore the letter case of characters by using the /I parameter.

Look for processes on Mailer1 with a CPU time of more than 30 minutes:

```
tasklist /s Mailer1 /fi "cputime gt 00:30:00"
```

Look for processes on Mailer1 that use more than 20,000 KB of memory:

```
tasklist /s Mailer1 /u adatum\wrstanek /fi "memusage gt 20000"
```

Enter multiple /Fi "Filter" parameters to specify that output must match against multiple filters:

```
tasklist /s Mailer1 /fi "cputime gt 00:30:00" /fi "memusage gt 20000"
```

Monitoring System Resource Usage and Processes

When you are working with processes, you'll often want to get a snapshot of system resource usage, which will show you exactly how memory is being used. One way to get such a snapshot is to use the Typeperf command to display current values for key counters of the memory object. As discussed in Chapter 6, "Event Logging, Tracking, and Monitoring," the Memory object is one of many performance objects available, and you can list its related performance counters by typing **typeperf -q Memory** at a command line.

Table 7-2 provides a summary of key counters of the Memory object. Most counters of the Memory object display the last observed value or the current percentage value rather than an average.

Table 7-2 Key Counters of the Memory Object

Memory Object Counter	Counter Description
% Committed Bytes In Use	The ratio of Committed Bytes to the Commit Limit. Committed memory is the physical memory in use for which space has been reserved in the paging file if it needs to be written to disk. The commit limit is determined by the size of the paging file. If Windows increases the paging file size, the commit limit increases as well, and the ratio is reduced.

Table 7-2 Key Counters of the Memory Object

Memory Object Counter	Counter Description
Available MBytes	The amount of physical memory, in megabytes, immediately available for allocation to a process or for system use. This is physical memory not currently being used and available for use. It is equal to the sum of memory assigned to the standby (cached), free, and zero page lists. When less than five percent of memory is free, the system is low on memory and performance can suffer.
Cache Bytes	The sum of the System Cache Resident Bytes, System Driver Resident Bytes, System Code Resident Bytes, and Pool Paged Resident Bytes counters. This provides information on the memory used by the operating system kernel. Critical portions of kernel memory must operate in physical memory and can't be paged to virtual memory; the rest of kernel memory can be paged to virtual memory.
Cache Bytes Peak	The maximum number of bytes used by the file system cache since the system was last restarted.
Cache Faults/sec	The rate at which faults occur when a page sought in the file system cache is not found and must be retrieved from elsewhere in memory (a soft fault) or from disk (a hard fault). The file system cache is an area of physical memory that stores recently used pages of data for applications.
Commit Limit	The amount of virtual memory, measured in bytes, that can be committed without having to extend the paging file(s). As the number of committed bytes grows, the paging file is allowed to grow up to its maximum size, which can be determined by subtracting the total physical memory on the system from the commit limit. If you set the initial paging file size too small, the system will repeatedly extend the paging file and this requires system resources. It is better to set the initial page size as appropriate for typical usage or simply use a fixed paging file size.
Committed Bytes	The amount of committed virtual memory, in bytes. Committed memory is the physical memory in use for which space has been reserved in the paging file if it needs to be written to disk. Each physical drive can have one or more paging files. If a system is using too much virtual memory relative to the total physical memory on the system, you might need to add physical memory.
Demand Zero Faults/sec	The rate at which a zeroed page is required to satisfy a fault, according to the difference between the values observed in the last two samples, divided by the duration of the sample interval. Pages emptied of previously stored data and filled with zeros are a security feature of Windows that prevent processes from seeing data stored by earlier processes that used the memory space.
Free & Zero Page List Bytes	The amount of physical memory, in bytes, that is assigned to the free and zero page lists. This memory does not contain cached data and is immediately available for allocation to a process or for system use.

Table 7-2 Key Counters of the Memory Object

Memory Object Counter	Counter Description
Free System Page Table Entries	The number of page table entries not currently in use by the system.
Modified Page List Bytes	The amount of physical memory, in bytes, that is assigned to the modified page list. Areas of memory on the modified page list contain cached data and code that is not actively in use by processes, the system, or the system cache. Windows needs to write out this memory before it will be available for allocation to a process or for system use.
Page Faults/sec	The number of pages faulted per second. This counter includes both hard and soft faults. Soft faults result in memory lookups. Hard faults require access to disk.
Page Reads/sec	The number of read operations required per second to resolve hard page faults. Hard page faults occur when a requested page isn't in memory and the computer has to go to disk to get it. Too many hard faults can cause significant delays and hurt performance.
Page Writes/sec	The number of page writes to disk to free up space in physical memory. Pages are written to disk only if they are changed while in physical memory.
Pages Input/sec	The rate at which pages are read from disk to resolve hard page faults. Hard page faults occur when a requested page isn't in memory and the computer has to go to disk to get it. Too many hard faults can cause significant delays and hurt performance.
Pages Output/sec	The rate at which pages are written to disk to free up space in physical memory. If the computer has to free up memory too often, this is an indicator that the system doesn't have enough physical memory (RAM).
Pages/sec	The number of memory pages that are read from disk or written to disk to resolve hard page faults. It is the sum of Pages Input/sec and Pages Output/sec.
Pool Non-paged Allocs	The number of calls to allocate space in the nonpaged pool. The nonpaged pool is an area of system memory area for objects that cannot be written to disk, and must remain in physical memory as long as they are allocated.
Pool Non-paged Bytes	The size, in bytes, of the nonpaged pool, an area of system memory for objects that cannot be written to disk, but must remain in physical memory as long as they are allocated. If the size of the nonpaged pool is large relative to the total amount of virtual memory allocated to the computer, you might want to increase the virtual memory size. If this value slowly increases in size over time, a kernel mode process might have a memory leak.

Table 7-2 Key Counters of the Memory Object

Memory Object Counter	Counter Description
Pool Paged Allocs	The number of calls to allocate space in the paged pool. The paged pool is an area of system memory for objects that can be written to disk when they are not being used.
Pool Paged Bytes	The total size, in bytes, of the paged pool. The paged pool is an area of system memory for objects that can be written to disk when they are not being used. If the size of the paged pool is large relative to the total amount of physical memory on the system, you might need to add memory to the system. If this value slowly increases in size over time, a kernel mode process might have a memory leak.
Pool Paged Resident Bytes	The size, in bytes, of the paged pool that is currently resident and actively being used. Typically, the resident bytes in the paged pool is a smaller amount than the total bytes assigned to the paged pool.
Standby Cache Core Bytes	The amount of physical memory, in bytes, that is assigned to the core standby cache page lists. A standby cached page list is an area of memory that contains cached data and code that is not actively in use by processes, the system, or the system cache. It is immediately available for allocation to a process or for system use. If the system runs out of available free–and-zero memory, memory on lower priority standby cache page lists will be repurposed before memory on higher priority standby cache page lists.
System Cache Resident Bytes	The size, in bytes, of the pageable operating system code in the file system cache. This value includes only current physical pages and does not include any virtual memory pages not currently resident.
System Code Resident Bytes	The size, in bytes of the operating system code currently in physical memory that can be written to disk when not in use.
System Code Total Bytes	The size, in bytes, of the pageable operating system code currently in virtual memory. It is a measure of the amount of physical memory being used by the operating system that can be written to disk when not in use and does not include code that must remain in physical memory and cannot be written to disk.
System Driver Resident Bytes	The size, in bytes, of the pageable physical memory being used by device drivers. It is the working set (physical memory area) of the drivers.
System Driver Total Bytes	The size, in bytes, of the pageable memory and pageable virtual memory currently being used by device drivers. It includes physical memory and code and data paged to disk.
Transition Faults/sec	The rate at which page faults are resolved by recovering pages that were being used by another process sharing the page, or were on the modified page list or the standby list, or were being written to disk at the time of the page fault. The pages were recovered without additional disk activity.

Table 7-2 Key Counters of the Memory Object

Memory Object Counter	Counter Description
Transition Pages RePurposed/sec	The rate at which the number of transition cache pages were reused for a different purpose. These pages would have otherwise remained in the page cache to provide a soft fault in the event the page was accessed in the future. These pages can contain private or sharable memory.
Write Copies/sec	The rate at which page faults are caused by attempts to write that have been satisfied by copying the page from elsewhere in physical memory.

Sample 7-1 provides an example of how you can use Typeperf to get a snapshot of memory usage. In this example, you use a counter file called Perf.txt to specify the counters you want to track. You collect five samples with an interval of 30 seconds between samples and save the output in a file called SaveData.txt. If you import the data into a spreadsheet or convert it to a table in a Microsoft Office Word document, you can make better sense of the output and will know exactly how the computer is using memory.

> **Note** I chose to track these counters because they give you a good overall snapshot of memory usage. If you save the command line as a script, you can run the script as a scheduled task to get a snapshot of memory usage at various times of the day.

Sample 7-1 Getting a snapshot of memory usage

Command-line

```
typeperf -cf c:\logs\perf.txt -o c:\logs\savedata.txt -sc 5 -si 30
```

Source for Perf.txt

```
\memory\% Committed Bytes In Use
\memory\Available MBytes
\memory\Cache Bytes
\memory\Cache Bytes Peak
\memory\Committed Bytes
\memory\Commit Limit
\memory\Page Faults/sec
\memory\Pool Nonpaged Bytes
\memory\Pool Paged Bytes
```

Sample output

```
"(PDH-CSV 4.0)","\\SERVER12\memory\% Committed Bytes In Use","
\\SERVER12\memory\Available MBytes","\\SERVER12\memory\Cache Bytes","
\\SERVER12\memory\Cache Bytes Peak","\\SERVER12\memory\Committed Bytes","
\\SERVER12\memory\Commit Limit","\\SERVER12\memory\Page Faults/sec","
\\SERVER12\memory\Pool Nonpaged Bytes","\\SERVER12\memory\Pool Paged
Bytes"
```

```
"03/25/2008 14:24:28.033","22.860837","2023.000000","260632576.000000","
280514560.000000","1636175872.000000","7157112832.000000","80.494007","
73240576.000000","152875008.000000"
"03/25/2008 14:24:30.033","22.861294","2023.000000","260653056.000000","
280514560.000000","1636208640.000000","7157112832.000000","70.997253","
73240576.000000","152875008.000000"
"03/25/2008 14:24:32.033","22.861294","2023.000000","260653056.000000","
280514560.000000","1636208640.000000","7157112832.000000","3.000142","
73261056.000000","152875008.000000"
"03/25/2008 14:24:34.033","22.861581","2023.000000","260673536.000000","
280514560.000000","1636229120.000000","7157112832.000000","15.999741","
73154560.000000","152875008.000000"
"03/25/2008 14:24:36.033","22.861695","2023.000000","260681728.000000","
280514560.000000","1636237312.000000","7157112832.000000","6.499981","
73134080.000000","152875008.000000"
```

You can obtain detailed information about running processes using the Windows PowerShell Get-Process cmdlet. See Table 7-3 for a summary of this cmdlet's significant properties. At a Windows PowerShell prompt, you can view important statistics for all processes by following these steps:

1. Get all the processes running on the server and store them in the *$a* variable by entering:

   ```
   $a =  get-process
   ```

2. Use the InputObject parameter to pass the process objects stored in *$a* to Get-Process and then pass the objects to the format-table cmdlet along with the list of properties you want to see by entering:

   ```
   get-process -inputobject $a | format-table –property ProcessName,
   BasePriority, HandleCount, Id, NonpagedSystemMemorySize,
   PagedSystemMemorySize, PeakPagedMemorySize, PeakVirtualMemorySize,
   PeakWorkingSet, SessionId, Threads, TotalProcessorTime,
   VirtualMemorySize, WorkingSet, CPU, Path
   ```

The order of the properties in the comma-separated list determines the display order. If you want to change the display order, simply move the property to a different position in the list.

When you know the process you want to examine, you don't need to use this multistep procedure. Simply enter the name of the process without the .exe or .dll instead of using –inputobject $a. In the following example, you list details about the winlogon process:

```
get-process winlogon | format-table –property ProcessName, BasePriority,
HandleCount, Id, NonpagedSystemMemorySize, PagedSystemMemorySize,
PeakPagedMemorySize, PeakVirtualMemorySize, PeakWorkingSet, SessionId,
Threads, TotalProcessorTime, VirtualMemorySize, WorkingSet, CPU, Path
```

You can enter part of a process name as well using an asterisk as a wildcard to match a partial name. In this example, Get-Process lists any process with a name that starts with winl:

```
get-process winl* | format-table –property ProcessName, BasePriority,
HandleCount, Id, NonpagedSystemMemorySize, PagedSystemMemorySize,
PeakPagedMemorySize, PeakVirtualMemorySize, PeakWorkingSet, SessionId,
Threads, TotalProcessorTime, VirtualMemorySize, WorkingSet, CPU, Path
```

Tip By default, many properties that measure memory usage are defined as 32-bit values. When working with Get-Process on 64-bit systems, you'll find that these properties have both a 32-bit and 64-bit version. On 64-bit systems with more than 4 GB of RAM, you'll need to use the 64-bit versions to ensure you get accurate values.

Table 7-3 Properties of Get-Process and How They Are Used

Property Name	Property Description
BasePriority	Shows the priority of the process. Priority determines how much of the system resources are allocated to a process. The standard priorities are Low (4), Below Normal (6), Normal (8), Above Normal (10), High (13), and Real-Time (24). Most processes have a Normal priority by default, and the highest priority is given to real-time processes.
CPU	Shows the percentage of CPU utilization for the process. The System Idle Process shows what percentage of CPU power is idle. A value of 99 for the System Idle Process means 99 percent of the system resources currently aren't being used. If the system has low idle time (meaning high CPU usage) during peak or average usage, you might consider upgrading to faster processors or adding processors.
Description	Shows a description of the process.
FileVersion	Shows the file version of the process's executable.
HandleCount	Shows the number of file handles maintained by the process. The number of handles used is an indicator of how dependent the process is on the file system. Some processes have thousands of open file handles. Each file handle requires system memory to maintain.
Id	Shows the run-time identification number of the process.
MinWorkingSet	Shows the minimum amount of working set memory used by the process.
Modules	Shows the executables and dynamically linked libraries used by the process.

Table 7-3 Properties of Get-Process and How They Are Used

Property Name	Property Description
NonpagedSystem-MemorySize / NonpagedSystem-MemorySize64	Shows the amount of virtual memory for a process that cannot be written to disk. The nonpaged pool is an area of RAM for objects that can't be written to disk. You should note processes that require a high amount of nonpaged pool memory. If the server doesn't have enough free memory, these processes might be the reason for a high level of page faults.
PagedSystem-MemorySize / PagedSystem-MemorySize64	Shows the amount of committed virtual memory for a process that can be written to disk. The paged pool is an area of RAM for objects that can be written to disk when they aren't used. As process activity increases, so does the amount of pool memory the process uses. Most processes have more paged pool than nonpaged pool requirements.
Path	Shows the full path to the executable for the process.
PeakPageMemorySize / PeakPageMemorySize64	Shows the peak amount of paged memory used by the process.
PeakVirtualMemorySize / PeakVirtualMemorySize64	Shows the peak amount of virtual memory used by the process.
PeakWorkingSet / PeakWorkingSet64	Shows the maximum amount of memory the process used, including both the private working set and the non-private working set. If peak memory is exceptionally large, this can be an indicator of a memory leak.
PriorityBoostEnabled	Shows a Boolean value that indicates whether the process has the PriorityBoost feature enabled.
PriorityClass	Shows the priority class of the process.
PrivilegedProcessorTime	Shows the amount of kernel-mode usage time for the process.
ProcessName	Shows the name of the process.
ProcessorAffinity	Shows the processor affinity setting for the process.
Responding	Shows a Boolean value that indicates whether the process responded when tested.
SessionId	Shows the identification number user (session) within which the process is running. This corresponds to the ID value listed on the Users tab in Task Manager.
StartTime	Shows the date and time the process was started.

Table 7-3 Properties of Get-Process and How They Are Used

Property Name	Property Description
Threads	Shows the number of threads that the process is using. Most server applications are multithreaded, which allows concurrent execution of process requests. Some applications can dynamically control the number of concurrently executing threads to improve application performance. Too many threads, however, can actually reduce performance, because the operating system has to switch thread contexts too frequently.
TotalProcessorTime	Shows the total amount of CPU time used by the process since it was started. If a process is using a lot of CPU time, the related application might have a configuration problem. This could also indicate a runaway or nonresponsive process that is unnecessarily tying up the CPU.
UserProcessorTime	Shows the amount of user-mode usage time for the process.
VirtualMemorySize / VirtualMemorySize64	Shows the amount of virtual memory allocated to and reserved for a process. Virtual memory is memory on disk and is slower to access than pooled memory. By configuring an application to use more physical RAM, you might be able to increase performance. To do this, however, the system must have available RAM. If it doesn't, other processes running on the system might slow down.
WorkingSet / WorkingSet64	Shows the amount of memory the process is currently using, including both the private working set and the non-private working set. The private working set is memory the process is using that cannot be shared with other processes. The non-private working set is memory the process is using that can be shared with other processes. If memory usage for a process slowly grows over time and doesn't go back to the baseline value, this can be an indicator of a memory leak.

Stopping Processes

When you want to stop processes that are running on a local or remote system, you can use the Taskkill command-line utility. With Taskkill, you can stop processes by process ID using the /Pid parameter or image name using the /Im parameter. If you want to stop multiple processes by process ID or image name, you can enter multiple /Pid or /Im parameters as well. With image names, however, watch out, because Taskkill will stop all processes that have that image name. Thus if three instances of Helpctr.exe are running, all three processes would be stopped if you use Taskkill with that image name.

As with Tasklist, Taskkill runs by default with the permissions of the user who is currently logged on, and you can also specify the remote computer whose tasks you

want to query, and the Run As permissions. To do this, you use the expanded syntax, which includes the following parameters:

```
/s Computer /u [Domain\]User [/p Password]
```

where *Computer* is the remote computer name or IP address, *Domain* is the optional domain name in which the user account is located, *User* is the name of the user account whose permissions you want to use, and *Password* is the optional password for the user account. If you don't specify the domain, the current domain is assumed. If you don't provide the account password, you are prompted for the password.

> **Note** Sometimes it is necessary to force a process to stop running. Typically, this is necessary when a process stops responding while opening a file, reading or writing data, or performing other read/write operations. To force a process to stop, you use the /F parameter. This parameter is only used with processes running on local systems. Processes stopped on remote systems are always forcefully stopped.

> **Tip** As you examine processes, keep in mind that a single application might start multiple processes. Generally, these processes depend on a central process, and from this main process a process tree containing dependent processes is formed. Occasionally, you may want to stop the entire process tree, starting with the parent application process and including any dependent processes. To do this, you can use the /T parameter.

Consider the following examples to see how you can use Taskkill:

Stop process ID 208:

```
taskkill /pid 208
```

Stop all processes with the image name Cmd.exe:

```
taskkill /im cmd.exe
```

Stop processes 208, 1346, and 2048 on MAILER1:

```
taskkill /s Mailer1 /pid 208 /pid 1346 /pid 2048
```

Force local process 1346 to stop:

```
taskkill /f /pid 1346
```

Stop a process tree, starting with process ID 1248 and including all child processes:

```
taskkill /t /pid 1248
```

To ensure that only processes matching specific criteria are stopped, you can use all the filters listed in Table 7-1 except SessionName. For example, you can use a filter to specify that only instances of Cmd.exe that are not responding should be stopped rather than all instances of Cmd.exe (which is the default when you use the /Im parameter).

As with Tasklist, Taskkill provides a Modules filter with operators EQ and NE to allow you to specify DLL modules that should be excluded or included. As you may recall, you use the Tasklist /M parameter to examine the relationship between running processes and DLLs configured on the system. Using the Taskkill Modules filter with the EQ operator, you could stop all processes using a specific DLL. Using the Taskkill Modules filter with the NE operator, you ensure that processes using a specific DLL are not stopped.

> **Tip** When you use filters, you don't have to specify a specific image name or process ID to work with. This means you can stop processes based solely on whether they match filter criteria. For example, you can specify that you want to stop all processes that aren't responding.

As with Tasklist, you can also use multiple filters. Again, you must use double quotation marks to enclose the filter string. Consider the following examples to see how you can use filters with Taskkill:

Stop instances of Cmd.exe that are not responding:

```
taskkill /im cmd.exe /fi "status eq not responding"
```

Stop all processes with a process ID greater than 4 if they aren't responding:

```
taskkill /fi "pid gt 4" /fi "status eq not responding"
```

Stop all processes using the Winspool.drv DLL:

```
taskkill /fi "modules eq winspool.drv"
```

Although the /Im and /Pid flags are not used in the second example, the process IDs are filtered so that only certain processes are affected. You don't want to stop the system or system idle process accidentally. Typically, these processes run with process IDs of 4 and 0 respectively, and if you stop them, the system will stop responding or shut down.

Detecting and Resolving Performance Issues Through Monitoring

At the command line, Tasklist and Windows PowerShell Get-Process provide everything you need for detecting and resolving most performance issues. However, you'll often need to dig deep to determine whether a problem exists and if so, what is causing the problem.

Monitoring Memory Paging and Paging to Disk

Often, you'll want to get detailed information on hard and soft page faults that are occuring. A page fault occurs when a process requests a page in memory and the system can't find it at the requested location. If the requested page is elsewhere in memory, the fault is called a soft page fault. If the requested page must be retrieved from disk, the fault is called a hard page fault.

To see page faults that are occurring in real time, enter the following at the command line:

```
typeperf "\memory\Page Faults/sec" -si 5
```

To stop Typeperf, press Ctrl+C. Page faults are shown according to the number of hard and soft faults occuring per second. Other counters of the Memory object that you can use for tracking page faults include:

- Cache Faults/sec
- Demand Zero Faults/sec
- Page Reads/sec
- Page Writes/sec
- Write Copies/sec
- Transition Faults/sec
- Transition Pages RePurposed/sec

Pay particular attention to the Page Reads/sec and Page Writes/sec, which provide information on hard faults. Although developers will be interested in the source of page faults, administrators are more interested in how many page faults are occurring.

Most processors can handle large numbers of soft faults. A soft fault simply means the system had to look elsewhere in memory for the requested memory page. With a hard fault, on the other hand, the requested memory page must be retrieved from disk, which can cause significant delays. If you are seeing a lot of hard faults, you may need to increase the amount of memory or reduce the amount of memory being cached by the system and applications.

In addition to counters of the Memory object discussed previously, you can use the following objects and counters to check for disk paging issues:

- **Paging File(*)\% Usage** The percentage of the paging file currently in use. If this value approaches 100 percent for all instances, you should consider either increasing the virtual memory size or adding physical memory to the system. This will ensure that the computer has additional memory if it needs it, such as when the computer load grows.
- **Paging File(*)\% Usage Peak** The peak size of the paging file as a percentage of the total paging file size available. A high value can mean that the paging file isn't large enough to handle increased load conditions.
- **PhysicalDisk(*)\% Disk Time** The percentage of time that the selected disk spent servicing read and write requests. Keep track of this value for the physical disks that have paging files. If you see this value increasing over several monitoring periods, you should more closely monitor paging file usage and you might consider adding physical memory to the system.
- **PhysicalDisk(*)\Avg. Disk Queue Length** The average number of read and write requests that were waiting for the selected disk during the sample interval. Keep track of this value for the physical disks that have paging files. If you see this value increasing over time and the Memory\Page Reads/Sec is also increasing, the system is having to perform a lot of paging file reads.

The asterisks in parentheses are placeholders for the object instance. If a particular object has multiple instances, such as when a computer has multiple physical disks or multiple paging files, you can use an object instance to track a specific occurrence of that object. You could also elect to track all instances of an object, such as whether you want to monitor all physical disks on a system. Specify _Total to work with all counter instances, or specify individual counter instances to monitor.

Sample 7-2 provides an example of how you can use Typeperf to get a snapshot of disk paging. In this example, you use a counter file called PagePerf.txt to specify the counters you want to track. You collect five samples with an interval of 30 seconds between samples and save the output in a file called SavePageData.txt. If you import the data into a spreadsheet or convert it to a table in a Word document, you can make better sense of the output and a better understanding of how the computer is using the page file and paging to disk.

Sample 7-2 Checking disk paging

Command line

```
typeperf -cf c:\logs\pageperf.txt -o c:\logs\savepagedata.txt -sc 5
-si 30
```

Source for PagePerf.txt

```
\memory\Pages/Sec
\Paging File(_Total)\% Usage
\Paging File(_Total)\% Usage Peak
\PhysicalDisk(_Total)\% Disk Time
\PhysicalDisk(_Total)\Avg. Disk Queue Length
```

Monitoring Memory Usage and the Working Memory Set for Individual Processes

You can use Tasklist to get basic memory usage for a process. The syntax you can use is:

```
tasklist /fi "pid eq ProcessID"
```

where *ProcessID* is the id number of the process you want to work with. The output from Tasklist will show you how much memory the process is currently using. For example, if you were tracking process ID 7292, your output might look like the following:

```
Image Name     PID      Session Name     Session#    Mem Usage
============== ======== ================ =========== ============
jvappm.exe     7292                              1      7,424 K
```

In this example, the process is using 7,424 KB of memory. By watching the memory usage over time, you can determine whether the memory usage is increasing. If memory usage is increasing compared to a typical baseline, the process might have a memory-related problem.

Sample 7-3 provides the source for a command-line script that checks the memory usage of a process over a timed interval. The script expects the process ID you want to work with to be passed as the first parameter. If you do not supply a process ID, error text is written to the output.

Sample 7-3 Viewing memory usage at the command line

MemUsage.bat

```
@echo off
if "%1"=="" (echo Error: please enter Process ID to track) & (goto
EXIT)

tasklist /fi "pid eq %1"
timeout /t 600
```

```
tasklist /fi "pid eq %1"
timeout /t 600
tasklist /fi "pid eq %1"
:EXIT
```

Sample output

```
Image Name                     PID Session Name        Session#    Mem Usage
========================= ======= ================ =========== ============
jvapm.exe                     7292                             1      7,452 K

Waiting for  0 seconds, press a key to continue ...

Image Name                     PID Session Name        Session#    Mem Usage
========================= ======= ================ =========== ============
jvapm.exe                     7292                             1      7,452 K

Waiting for  0 seconds, press a key to continue ...

Image Name                     PID Session Name        Session#    Mem Usage
========================= ======= ================ =========== ============
jvapm.exe                     7292                             1      7,452 K
```

In Sample 7-3, the process's memory usage does not change over the sampled interval. Because of this, it is unlikely the process has a memory leak, but to be sure you'd need to sample over a longer period.

You can use the Windows PowerShell Get-Process cmdlet to track detailed memory usage for individual processes. The syntax you can use is

```
get-process ProcessName | format-table –property
NonpagedSystemMemorySize, PagedSystemMemorySize, VirtualMemorySize,
PeakVirtualMemorySize, MinWorkingSet, WorkingSet, PeakWorkingSet
```

where *ProcessName* is the name of the process without the .exe or .dll. In a Windows PowerShell script, such as the one shown as Sample 7-4, you could combine the Get-Process cmdlet with the start-sleep cmdlet to view the memory usage for a process at timed intervals.

Sample 7-4 Viewing detailed memory usage

MemUsage.ps1

```
get-process msdtc | format-table –property NonpagedSystemMemorySize,
PagedSystemMemorySize, VirtualMemorySize, PeakVirtualMemorySize,
MinWorkingSet, WorkingSet, PeakWorkingSet

start-sleep –seconds 600

get-process msdtc | format-table –property NonpagedSystemMemorySize,
PagedSystemMemorySize, VirtualMemorySize, PeakVirtualMemorySize,
MinWorkingSet, WorkingSet, PeakWorkingSet
```

```
start-sleep –seconds 600

get-process msdtc | format-table –property NonpagedSystemMemorySize,
PagedSystemMemorySize, VirtualMemorySize, PeakVirtualMemorySize,
MinWorkingSet, WorkingSet, PeakWorkingSet
```

Sample output

```
   Nonpaged PagedSystem     Virtual       Peak    Working       Peak
     System      Memory      Memory    Virtual        Set    Working
 MemorySize        Size        Size MemorySize                   Set
----------- ----------- ----------- ---------- ---------- ----------
       6304       70544    41766912   63631360    6287360    6344704

   Nonpaged PagedSystem     Virtual       Peak    Working       Peak
     System      Memory      Memory    Virtual        Set    Working
 MemorySize        Size        Size MemorySize                   Set
----------- ----------- ----------- ---------- ---------- ----------
       8123       96343    56243535   97423424    9147256    9348942

   Nonpaged PagedSystem     Virtual       Peak    Working       Peak
     System      Memory      Memory    Virtual        Set    Working
 MemorySize        Size        Size MemorySize                   Set
----------- ----------- ----------- ---------- ---------- ----------
      17564      129645    48934246   97423424    9987384   10344706
```

Note Windows PowerShell script files have the .ps1 filename extension. To run a script at the Windows PowerShell prompt, you type the name of the script and, optionally, the filename extension. You must specify the fully qualified path to the script file, even if the script is in the current directory. To indicate the current directory, type the directory name or use the dot (.) to represent the current directory. With the MemUsage.ps1 script in the current directory, you can run the script by entering **.\memusage.ps1** at the Windows PowerShell prompt.

The Get-Process properties examined in Sample 7-4 provide the following information:

- **NonPagedSystemMemorySize** Shows the amount of allocated memory that can't be written to disk
- **PagedSystemMemorySize** Shows the amount of allocated memory that is allowed to be paged to the hard disk
- **VirtualMemorySize** Shows the amount of virtual memory allocated to and reserved for a process
- **PeakVirtualMemorySize** Shows the peak amount of paged memory used by the process
- **WorkingSetSize** Shows the amount of memory allocated to the process by the operating system
- **PeakWorkingSet** Shows the peak amount of memory used by the process

When you focus on these properties, you are zeroing in on the memory usage of a specific process. The key aspect to monitor is the working memory set. The working set of memory shows how much memory is allocated to the process by the operating system. If the working set increases over time and doesn't eventually go back to baseline usage, the process may have a memory leak. With a memory leak, the process isn't properly releasing memory that it's using, which can lead to reduced performance of the entire system.

In Sample 7-4, the process's memory usage changes substantially over the sampled interval. While it is most likely the process is simply actively being used by users or the computer itself, the process should eventually return to a baseline memory usage. If this doesn't happen, the process may have a memory-related problem.

Resolving Performance Bottlenecks

Because memory is usually the primary performance bottleneck on both workstations and servers, I've discussed many techniques previously in this chapter that you can use to help identify problems with memory. Memory is the resource you should examine first to try to determine why a system isn't performing as expected.

However, memory isn't the only bottleneck. Processor bottlenecks can occur if a process's threads need more processing time than is available. If a system's processors are the performance bottleneck, adding memory, drives, or network connections won't solve the problem. Instead, you might need to upgrade the processors to faster clock speeds or add processors to increase the computer's upper capacity. With servers, you could also move processor-intensive applications to another server.

Typeperf counters you can use to check for processor bottlenecks include the following:

- **System\Processor Queue Length** Records the number of threads waiting to be executed. These threads are queued in an area shared by all processors on the system. The processor queue grows because threads have to wait to get processing time. As a result, the system response suffers and the system appears sluggish or nonresponsive. Here, you might need to upgrade the processors to faster clock speeds or add processors to increase the server's upper capacity.
- **Processor(*)\% Processor Time** Records the percentage of time the selected processor is executing a non-idle thread. You should track this counter separately for each processor instance on the server. If the % Processor Time values for all instances are high (above 75 percent) while the network interface and disk input/output (I/O) throughput rates are relatively low, you might need to upgrade the processors to faster clock speeds or add processors to increase the server's upper capacity.
- **Processor(*)\% User Time** Records the percentage of time the selected processor is executing a non-idle thread in User mode. *User mode* is a processing mode for applications and user-level subsystems. A high value for all processor instances

might indicate that you need to upgrade the processors to faster clock speeds or add processors to increase the server's upper capacity.

- **Processor(*)\% Privileged Time** Records the percentage of time the selected processor is executing a non-idle thread in Privileged mode. *Privileged mode* is a processing mode for operating system components and services, allowing direct access to hardware and memory. A high value for all processor instances might indicate that you need to upgrade the processors to faster clock speeds or add processors to increase the computer's upper capacity.
- **Processor(*)\Interrupts/sec** Records the average rate, in incidents per second, that the selected processor received and serviced hardware interrupts. If this value increases substantially over time without a corresponding increase in activity, the system might have a hardware problem. To resolve this problem, you must identify the device or component that is causing the problem. Each time drivers or disk subsystem components such as hard disk drives or network components generate an interrupt, the processor has to stop what it is doing to handle the request because requests from hardware take priority. However, poorly designed drivers and components can generate false interrupts, which tie up the processor for no reason. System boards or components that are failing can generate false interrupts as well.

Note The asterisks in parentheses are placeholders for the object instance. On multiprocessor systems, you might need to rule out processor affinity as a cause of a processor bottleneck. By using processor affinity, you can set a program or process to use a specific processor to improve its performance. Assigning processor affinity can, however, block access to the processor for other programs and processes.

A system's hard disks are rarely the primary reason for a bottleneck. If a system is having to do a lot of disk reads and writes, it is usually because there isn't enough physical memory available and the system has to page to disk. Because reading from and writing to disk is much slower than reading and writing memory, excessive paging can degrade the server's overall performance. To reduce the amount of disk activity, you want the system to manage memory as efficiently as possible and page to disk only when necessary.

You can use these counters to monitor disk reads and writes:

- **PhysicalDisk(*)\% Disk Time** Records the percentage of time the physical disk is busy. Track this value for all hard disk drives on the system in conjunction with Processor(*)\% Processor Time and Network Interface(*)\Bytes Total/sec. If the % Disk Time value is high and the processor and network connection values aren't high, the system's hard disk drives might be creating a bottleneck.
- **PhysicalDisk(*)\Current Disk Queue Length** Records the number of system requests that are waiting for disk access. A high value indicates that the disk-waits are impacting system performance. In general, you want very few waiting requests.

- **PhysicalDisk(*)\Avg. Disk Write Queue Length** Records the number of write requests that are waiting to be processed.
- **PhysicalDisk(*)\Avg. Disk Read Queue Length** Records the number of read requests that are waiting to be processed.
- **PhysicalDisk(*)\Disk Writes/sec** Records the number of disk writes per second, which indicates the amount of disk I/O activity. By tracking the number of writes per second and the size of the write queue, you can determine how write operations are impacting disk performance.
- **PhysicalDisk(*)\Disk Reads/sec** Records the number of disk reads per second, which indicates the amount of disk I/O activity. By tracking the number of reads per second and the size of the read queue, you can determine how read operations are impacting disk performance.

Networking components can also cause bottlenecks. A delay between when a request is made, the time the request is received, and the time a user gets a response can cause users to think that systems are slow or nonresponsive. Unfortunately, in many cases, the delay users experience when working over the network is beyond your control. This is because the delay is a function of the type of connection the user has and the route the request takes. The total capacity of a computer to handle requests and the amount of bandwidth available to a computer are factors you can control, however. Network capacity is a function of the network cards and interfaces configured on the computers. Network bandwidth availability is a function of the network infrastructure and how much traffic is on it when a request is made.

You can use the following counters to check network activity and look for bottlenecks:

- **Network Interface(*)\Bytes Received/Sec** Records the rate at which bytes are received over a network adapter.
- **Network Interface(*)\Bytes Sent/Sec** Records the rate at which bytes are sent over a network adapter.
- **Network Interface(*)\Bytes Total/Sec** Records the rate at which bytes are sent and received over a network adapter. Check the network card configuration if you think there's a problem.
- **Network Interface(*)\Current Bandwidth** Estimates the current bandwidth for the selected network adapter in bits per second. Check to ensure the current bandwidth matches the type of network card configured on the computer. Most computers use 10 megabit, 100 megabit, or 1 gigabit network cards. Keep in mind that if the computer has a 1 gigabit network card, the networking devices to which the computer connects must also support this speed.

Index

Symbols and Numbers

A

B

C

D

E

F

G

H

I

N

O

P

Q

S

T

X

About the Author

William R. Stanek (*http://www.williamstanek.com/*) has more than 20 years of hands-on experience with advanced programming and development. He is a leading technology expert, an award-winning author, and a pretty-darn-good instructional trainer. Over the years, his practical advice has helped millions of programmers, developers, and network engineers all over the world. He has written more than 75 books. Current or forthcoming books include *Microsoft Exchange Server 2007 Administrator's Pocket Consultant*, Second Edition, *Windows Server 2008 Administrator's Pocket Consultant, Microsoft SQL Server 2008 Administrator's Pocket Consultant*, and *Windows Server 2008 Inside Out*.

William has been involved in the commercial Internet community since 1991. His core business and technology experience comes from more than 11 years of military service. He has substantial experience in developing server technology, encryption, and Internet solutions. He has written many technical white papers and training courses on a wide variety of topics. He frequently serves as a subject matter expert and consultant.

William has an MS with distinction in information systems and a BS in computer science, magna cum laude. He is proud to have served in the Persian Gulf War as a combat crewmember on an electronic warfare aircraft. He flew on numerous combat missions into Iraq and was awarded nine medals for his wartime service, including one of the United States of America's highest flying honors, the Air Force Distinguished Flying Cross. Currently he resides in the Pacific Northwest with his wife and children.