

**Windows**  
& .NET MAGAZINE

**eBooks**

**Windows**  
& .NET MAGAZINE

**TECHNICAL REFERENCE**

**Winternals**

A Guide to

# **Windows** **Power** **Tools**

**Mark Minasi**

**Windows & .NET Magazine Technical Reference**

# **A Guide to Windows Power Tools**

*By Mark Minasi*

**Windows**  
& .NET MAGAZINE

A Division of Penton Media

# Windows

& .NET MAGAZINE

Copyright 2003

*Windows & .NET Magazine*

All rights reserved. No part of this book may be reproduced in any form by an electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

It is the reader's responsibility to ensure procedures and techniques used from this book are accurate and appropriate for the user's installation. No warranty is implied or expressed.

ISBN 1-58304-513-9

## About the Author

**Mark Minasi** (<http://www.minasi.com/gethelp>) is a contributing editor for *Windows & .NET Magazine*, an MCSE, and the author of *Mastering Windows 2000 Server*, 4th Edition (Sybex). He writes and speaks around the world about Win2K and Windows NT networking.



## Table of Contents

<b>Introduction</b> .....	<b>viii</b>
<b>Chapter 1: Computer Management Tools</b> .....	<b>1</b>
Cachemov .....	1
Con2prt .....	1
Delprof .....	2
Profiles' Problems .....	2
Delete Remote Profiles .....	2
Clean Up Your Local Hard Disk .....	3
Elogdmp .....	3
Instmon .....	4
The Instmon Trio .....	5
Putting the Trio to the Test .....	5
Movetree .....	6
Regfind .....	7
Srvany .....	8
Tlist and Kill .....	10
Uptime .....	12
<b>Chapter 2: Desktop Tools</b> .....	<b>13</b>
Clear Screen Saver .....	13
Cliptray .....	13
Cmdhere .....	14
Runext .....	15
TweakUI .....	15
TweakUI Tabs .....	16
Winexit .....	16
Winexit Options .....	17
<b>Chapter 3: Diagnostic Tools</b> .....	<b>19</b>
Browser Monitor and Domain Monitor .....	19
Jewel of a Tool .....	19
Monitor Mystery .....	19
Driver Verifier .....	20
OH .....	21

<b>Chapter 4: File and Disk Tools</b> . . . . .	<b>23</b>
Compress . . . . .	23
Fileman.vbs . . . . .	23
Freedisk . . . . .	24
Permcopy and Share.vbs . . . . .	25
Xcopy . . . . .	26
<b>Chapter 5: Network Management Tools</b> . . . . .	<b>28</b>
Addusers . . . . .	28
Creating an Account . . . . .	28
Addusers' Options . . . . .	28
Browstat . . . . .	29
Cusmgr . . . . .	30
Dhccpmd . . . . .	31
The Problem . . . . .	31
Problem Solved . . . . .	32
FAZAM 2000 RFV . . . . .	33
Logoff.exe and Logoff.vbs . . . . .	34
Netdom for Windows NT . . . . .	35
Netdom's Member Option . . . . .	36
Adding Accounts to a Domain . . . . .	36
Other Netdom Member Commands . . . . .	37
Netdom 2000 . . . . .	37
Netdom 2000's Trust . . . . .	38
Nltest . . . . .	39
Rcmd . . . . .	41
Repadmin . . . . .	42
Get a Handle on AD Internals . . . . .	42
Forcing AD Replication . . . . .	43
Tracking AD Replication . . . . .	45
AD Replication and Up-to-Date Vectors . . . . .	46
Rmtshare . . . . .	47
Setprfdc . . . . .	48
ShareUI . . . . .	49
A Problem Finding Shares . . . . .	49
Solution: ShareUI . . . . .	50
A Fly in the Soup . . . . .	50
In Short . . . . .	50
Showmbrs and Diskquotas.pl . . . . .	50
SU . . . . .	52
When this Utility Is Useful . . . . .	52
Step by Step . . . . .	52
What Can Go Wrong . . . . .	52

**Chapter 5: Network Management Tools (continued)**

Telnet Server	.53
Installing Telnet's Services	.53
Put to the Test	.54
Proceed with Caution	.54
Winscl	.54
Winscl Commands	.55
Xcacls	.56

**Chapter 6: Scripting Tools . . . . .58**

AutoExNT	.58
How to Make It Work	.58
Interactive AutoExNT	.59
Dnsserver.pl, Dnszones.pl, and Dnsrecord.pl	.59
Ifmember and KiXtart	.60
Logevent	.61
Munge	.63
Service.vbs	.64
VBScript Tools in Supplement 4	.65
WinAT, Soon, and Sleep	.67

# Introduction

Windows 2000 and Windows NT provide a powerful set of graphical tools that network administrators can use to manage network servers and desktop systems. Although Win2K's and NT's built-in graphical tools are powerful and easy to use, they can be cumbersome and even error-prone for highly repetitive tasks. In addition, the tools don't lend themselves to scripting. Fortunately, Microsoft realized early on that network administrators needed a set of powerful and scriptable tools to manage servers and desktop systems from the command line. In response to this need Microsoft created a set of resource kits for the company's Windows Server and Windows Professional editions. Microsoft's Windows resource kits contain a wealth of tools that let you perform most Windows administrative tasks from the command line. In addition, you can usually incorporate these commands into your own management scripts. The resource kits aren't included in the Windows products; you must purchase the resource kits separately. You can find more information about the resource kits at <http://www.microsoft.com/windows2000/techinfo/reskit>. In addition to the resource kits, Microsoft added an extra set of tools called Windows Support Tools to the Windows installation CD-ROM. The Windows Support Tools folder is on the Windows Server and Windows Professional CD-ROMs in the \support folder.

This book will provide you with a practical introduction to some of the most important tools in the resources kits and the Support Tools. Each chapter presents a different group of tools as they relate to various network management tasks. The tools in Chapter 1 let you perform computer management tasks such as deleting profiles, dumping the event log, finding registry entries, and killing processes. This chapter includes information about using the Movetree tool, which can move Active Directory (AD) objects between organizational units (OUs). Chapter 2 focuses on desktop productivity tools. Here you'll learn about tools such as Cmdhere, which opens a command window starting in a specified directory, and TweakUI, which can customize several of your desktop settings. Chapter 3 presents a group of network diagnostic tools such as Browser Monitor, which queries your network's browser list, and Domain Monitor, which queries domain controllers (DCs) on your network. In Chapter 4 you'll see how to use several file and disk tools. You'll learn how to use the Compress command to save disk space and how to use the share.vbs script to create shares. This chapter also presents the fileman.vbs script, which you can use to change the ownership of files on an NTFS volume. Chapter 5 presents some important network management tools. In this chapter you'll learn how to use the Addusers command to create new user accounts and how to use the Netdom tool to manage domains. This chapter includes information about using the Dhcpcmd tool, which lists active DHCP leases. In Chapter 6 you'll see how to use some scripting tools. Among other things, this chapter presents a group of Perl scripts that can perform virtually all DNS-related tasks. This chapter also presents the Logevent command, which you can use to write events into the event log for your own batch files.

You'll find this book to be a valuable and practical guide that enables you to quickly master the most important tools in the resource kits and the Support Tools.

## Chapter 1

# Computer Management Tools

## Cachemov

The *Microsoft Windows 2000 Server Resource Kit Supplement One's* Cachemov tool works with Win2K's helpful Offline Files feature. Offline Files lets you cache network files on your local hard disk. When you want to work with a cached network file, Offline Files quickly checks whether the server's copy of the file has changed. If it hasn't, your system delivers the file from the local cache, which is far faster than retrieving the file from the network.

I locally cache all the files that I frequently use. But those files are numerous and contain a lot of data, and Offline Files keeps all of it on the same hard disk that contains the OS. On my system, that hard disk is fairly full, so I added a hard disk to hold the cached files and called the new hard disk drive D.

But how did I tell Offline Files to cache my files on the new drive? I simply typed Cachemov at a command line, and a prompt asked me what drive I wanted Offline Files to use. I could have typed

```
cachemov -unattend d:\
```

to use the command in unattended mode.

Now, my data is on a physically secure server, which I regularly back up, and my server's battery backup protects the data. But I also have local copies of the data—just in case the server is temporarily unavailable—and plenty of space for those copies.

## Con2prt

Con2prt lets you add a network printer to a workstation, make a network printer the default printer, or clear all printers from the workstation. To make a network printer named \\server1\hplaser the default printer for the workstation you're sitting at, type

```
con2prt /cd \\server1\hplaser
```

To add the printer without making it the default, drop the *d* from the command string. To clear all printers from the workstation, use the */f* switch instead of */cd* or */c*.

You can use a Telnet session to run Con2prt on a remote workstation. The tool also works in batch files. For example, a batch file that contains the statements

```
con2prt /f
con2prt /cd \\server1\printer1
con2prt /c \\server1\printer2
```

will delete all existing printer connections, add Printer1 as the default printer, then add Printer2.

You'll find Con2prt in the *Microsoft Windows 2000 Resource Kit* and in the Zero Administration Kit (ZAK) for Windows (available at <http://www.microsoft.com/ntworkstation/downloads/recommended/featured/ntzak.asp>). To use the tool, you need only the con2prt.exe file; no complex installation is required.

### Delprof

User profiles in Windows NT 4.0 are collections of user-specific data, including desktop settings, the Programs menu, application settings (such as whether Microsoft Word hyphenates words), and the contents of the Send To menu that pops up when you right-click items on the NT desktop. If you store user profiles on a network share, every time users log on to a computer on the network their profile follows them.

### Profiles' Problems

These roaming profiles are useful, but they can create clutter on network workstations. Suppose I'm an administrator with a roaming profile and I log on to the network from your workstation to fix a problem. NT downloads my entire profile to your hard disk and leaves my profile on your hard disk after I log off.

This residual profile is troublesome for two reasons. First, it consumes space on your hard disk. Second, it might include sensitive information that I don't want you to access. For example, if I write myself a note in Notepad and leave the note on my desktop, my profile includes a copy of the note. When I log off your workstation, the note remains on your hard disk. If your computer has an NTFS-formatted system disk, NTFS permissions protect access to the profile. But if the disk is FAT-formatted, nothing prevents you from reading my note.

As an administrator, I can change your computer's registry to tell NT not to keep local copies of roaming profiles on your machine after the profile's owner logs off. If I go to HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon, add the DeleteRoamingCache value entry (type REG\_DWORD), and set DeleteRoamingCache to 1, your workstation won't keep my profile after I log off.

But if you have a roaming profile, your workstation won't keep your profile after you log off either; so, this registry change will force you to wait for the profile to download every time to log on to your workstation. You probably won't thank me for making this change.

I need to remove my profile from your machine without affecting the way your workstation treats your profile. And, I can't simply go to the User Profiles tab on the Control Panel System applet to remove my profile before I log off your computer because I can't remove the profile I'm logged on under.

### Delete Remote Profiles

To remove my profile from your hard disk, I need a remote profile zapper, a program I can run when I get back to my desk that erases the copy of my profile from your hard disk. *Microsoft Windows NT Server 4.0 Resource Kit* provides just such a tool, Delprof.

The syntax for Delprof is

```
delprof [/p] /c:\\<remotecomputername>
```

The /p option tells Delprof to prompt me before it deletes any profiles from your machine. If your computer's name is GOLDEN, I type

```
delprof /p /c:\\golden
```

on my computer to remove my profile from your hard disk. NT then shows me all GOLDEN's profiles, one at a time, and asks whether I want to delete each profile. Here's a sample dialog:

```
Delete \\GOLDEN\admin$\Profiles\Administrator? (Yes/No/All) n
Delete \\GOLDEN\admin$\Profiles\NormUser? (Yes/No/All) n
Delete \\GOLDEN\admin$\Profiles\Mark? (Yes/No/All) y
Deleting \\GOLDEN\admin$\Profiles\Mark... [Ok]
```

## Clean Up Your Local Hard Disk

Delprof can also easily remove multiple profiles from one computer. When I try out new programs on my NT workstation, I create user accounts that I use only once. After a while, my Profiles directory contains a lot of profiles that I haven't used in months.

I could use the Control Panel System applet to delete these accounts one by one, but Delprof offers a better solution:

```
delprof [/q] /d:<numdays>
```

This command deletes all profiles that no one has touched in *numdays* days. The /q option tells Delprof to complete the deletions without prompting me for responses. Delprof is an easy solution to the tricky problem of removing unneeded roaming profiles.

## Elogdmp

Windows 2000 and Windows NT administrators quickly learn the value of the event logs. If you don't look over the event logs every day or two, you might miss some important warnings or errors. But starting up the Microsoft Management Console (MMC) Computer Management snap-in is a pain—you're more likely to scan event logs regularly if you have a command-line tool.

Event Log Query Tool (Elogdmp), an automated tool for examining event logs, is available in the *Microsoft Windows 2000 Server Resource Kit Supplement One*. Elogdmp is a bit limited, but with some work on your part, it can be quite useful. The tool dumps an event-log summary to the screen. You can then search the output for particular keywords or pipe it into a batch file for processing.

Elogdmp's syntax is

```
elogdmp <computername> <logname>
```

*Computername* is the name of the computer whose log you want to dump—Elogdmp takes Net-BIOS names, IP addresses, and some (but not all) DNS names (preceding backslashes aren't required). *Logname* is the name of the log that you want to see (e.g., Security, Application, Directory Service). Case doesn't seem to matter, but if the log name contains a space, enclose the name in quotation marks.

## 4 A Guide to Windows Power Tools

So, for example, to dump the System log on a machine named \\mypc, you'd type

```
eologdmp mypc system
```

If \\mypc is a DNS server, you can type

```
eologdmp mypc "dns server"
```

to view the DNS Server log. The output is a comma-delimited dump of the log's contents: the date, time, source, type, category, event ID, user, and machine information. Most of the fields are self-explanatory, but the source, type, and category fields need some explanation.

The source field identifies the program (e.g., w32time, DNS) that created the log entry, and the type field contains the severity level (i.e., INFO, WARN, or ERROR). In a log file you view using the Computer Management snap-in, the category field provides more specific information about the system than the source field does—for example, an Active Directory (AD)-related message announcing that the OS has begun defragmenting AD might display NTDS ISAM as the source and the more helpful Online Defragmentation as the category. Unfortunately, Elogdmp's category field is singularly unhelpful: It contains only the word None or Something.

Although Elogdmp doesn't display the actual message that triggered the log entry, the tool still can be useful from the command line or in a batch file. For example

```
eologdmp mypc system | find "ERROR"
```

will show all System event-log entries of type ERROR (the Find command is case sensitive, so ERROR must be in capital letters). The entries appear from the oldest to the most recent, so you'll see today's errors even if the bulk of the output scrolls off the screen.

Elogdmp is terse in its error reporting. If it can't satisfy your request (e.g., if you query an event log on a machine that you don't have permissions to), you'll get a decidedly unhelpful error such as

```
eologdmp: cannot open the 'system' event log (5)
```

To determine the reason for the error, you need to know that the number in parentheses is the number of the error. Error 5 happens to be an access-denied error. To see what an error means, you can use the Net Helpmsg command. For example, typing

```
net helpmsg 5
```

at a command prompt produces the response *access is denied*.

### Instmon

Remember the good old days when all you needed to do to uninstall an application was delete its directory and remove the directory from the Path command? Nowadays, application installers don't just create directories and files; they modify the registry, profiles, and who knows what else.

These invasive installations leave administrators asking, "What exactly did this program do when it installed?" More important, administrators wonder, "What must I do to completely uninstall the application?"

Sure, many programs come with uninstallation programs, but these uninstallers are of limited value. They usually don't remove registry settings, and they often leave directories and files behind. Windows NT administrators need a set of utilities that can spy on a setup program, note every change the program makes, and use that information to undo the installation.

### ***The Instmon Trio***

The *Microsoft Windows NT Server 4.0 Resource Kit Supplement Two* comes with a trio of utilities—Instaler, Showinst, and Undoinst—that work together to completely remove applications from NT. After you install the resource kit, you find the uninstallation utilities in the Instmon subdirectory of the directory you installed the resource kit utilities in.

Instaler is the trio's spy. Instead of directly running a setup program, you tell Instaler to run the setup when you want to monitor an application's installation. Instaler records everything the setup program does, including obscure operations such as making API calls and debugging output. Instaler records the setup program's changes to your system in a non-ASCII file with the extension .iml. The utility saves the .iml file in the Instmon directory.

Showinst is the tattler. It reads the .iml file that Instaler creates and reports to you the information that Instaler records.

Undoinst is the fixer. It uses the information from the .iml file to ruthlessly zap files, directories, and registry entries.

When Undoinst completes, the application is gone, and your system contains almost no trace of it. (The only trace of files that Undoinst has trouble removing is applications' listing on your Programs menu. Hey, no utility's perfect.)

### ***Putting the Trio to the Test***

To try the Instmon utilities, install a 32-bit program such as winzip95.exe. WinZip 6.3 is a good 32-bit file compression and uncompression routine, and winzip95.exe is WinZip's setup routine.

To use Instaler to install WinZip, type

```
instaler wzip winzip95.exe
```

at a command prompt. (Wzip is the filename I selected for the .iml and log files Instaler creates during the WinZip installation.) Instaler displays hundreds of lines of information about what winzip95.exe is doing. You can safely ignore this information as it scrolls by, but if you'd like to look at the information in detail after the installation completes, open the wzip.log file. (You don't need wzip.log to uninstall WinZip.)

After you enter the instaler command, winzip95.exe runs, installing WinZip. After WinZip installs, Instaler terminates, telling you it created a file called wzip.iml.

If you want to find out the names of the files winzip95.exe created and see what other changes the WinZip installation made, type

```
showinst wzip >report.txt
```

at a command prompt. This command will generate a text file, report.txt, that summarizes all the file, directory, registry, and .ini file changes that winzip95.exe made to your system.

If you want to wipe WinZip off your computer altogether, use the application terminator, Undoinst. Open a command prompt and type

```
undoinst wzip
```

In my tests, Undoinst removed all traces of WinZip except for its presence on my Programs menu. I still see a WinZip entry when I click Start, Programs, even though the application is no longer on my machine. The Instmon programs don't offer the functionality of Zero Administration for Windows (ZAW), but they're pretty neat!

### Movetree

I sometimes feel that I never design anything correctly the first time, which is why I like the Movetree tool. Movetree is a command-line utility that lets you move user accounts, computer accounts, groups, and organizational units (OUs) from one domain in an Active Directory (AD) forest to another domain in the same forest.

You use the following syntax with Movetree: `movetree operation /s source domain controller /d destination domain controller /sdn source distinguished name /ddn destination distinguished name`. The *operation* field contains the command `/check`, `/start`, or `/continue`. The `/check` command runs a test to see whether your Movetree command syntax makes sense and to ensure that you have permission to perform the action you're trying to complete. To move an object from one domain to another, you need to log on with an account that both domains recognize as an administrator account. The `/check` option doesn't catch every problem; I've run `/check` and found no problems but had the Movetree command fail when I tried to run it. The `/start` option runs the Movetree command. The `/continue` option restarts the Movetree command if the command stops before it completes (e.g., if it encounters a problem partway through the operation). This switch lets Movetree restart from a partially completed move rather than starting from scratch.

The *source domain controller* and *destination domain controller* options designate the DNS names of a domain controller (DC) from the source and destination domains. Suppose you're running the pepsi.com and tacobell.com domains, and the tacobell.com domain has an OU named gorditas that markets some of your company's products. Then, suppose Pepsi wants to centralize the marketing folks in all the company's divisions, including Taco Bell, into the pepsi.com domain. To write a Movetree command to accomplish this task, you need to know the names of one of the pepsi.com DCs and one of the tacobell.com DCs. For illustrative purposes, let's call the DCs pepsgen.pepsi.com and bigbell.tacobell.com.

The *source distinguished name* option is the name (e.g., gorditas) and location (e.g., tacobell.com) of the object you want to move. The *destination distinguished name* is the name of the object (e.g., gorditas) and the location you want to move the object to (e.g., pepsi.com). To complete the Movetree command in my example, you specify that the command is moving an OU named gorditas from tacobell.com to pepsi.com.

The tricky part about writing the source distinguished name is that you need to use a distinguished name (DN) format to specify an OU named gorditas in tacobell.com. In DN format, you'd write `ou=gorditas,dc=tacobell,dc=com`. The `ou` parameter is the name of the organizational unit, and the `dc` parameter is the device context. You use as many `dc` parameters in the DN as necessary to express the parts of the DNS name. For example, the DN for the domain downtown.acme.com is `dc=downtown,dc=acme,dc=com`. You use the parameter `cn` (common

name) to designate objects that aren't OUs and aren't part of the DNS name (e.g., user accounts, groups, non-OU folders such as Users or Computers). Thus, the source distinguished name for an account named Sue in the Users folder of tacobell.com is `cn=sue,cn=Users,dc=tacobell,dc=com`.

The Movetree command to move the OU `gorditas` from the domain `tacobell.com` to the domain `pepsi.com` is `movetree /start /s bigbell.tacobell.com /d pepsigen.pepsi.com /sdn ou=gorditas,dc=tacobell,dc=com /ddn ou=gorditas,dc=pepsi,dc=com`. Suppose you have a user named Larry, whose account you want to move from an OU named `burritos` in `tacobell.com` to the Users folder in `pepsi.com`. The Movetree command to accomplish this task is `movetree /start /s bigbell.tacobell.com /d pepsigen.pepsi.com /sdn cn=larry,ou=burritos,dc=tacobell,dc=com /ddn cn=larry,cn=Users,dc=pepsi,dc=com`.

Movetree has several limitations. You can't use the command to consolidate users, computers, or OUs from one forest to another. In addition, you can't move part of a group—you must move all the group's members. For example, if you want to move a group from domain A to domain B, but the group has members in domain A whose user accounts aren't moving to domain B, you can't use Movetree to move the group. Finally, Movetree works only on AD domains running in native mode (i.e., with no Windows NT 4.0 DCs). The command's documentation neglects to mention this important point.

## Regfind

Every Windows geek knows that you can use the registry to fix just about anything. The trick is finding the particular registry subkey that you need. On most occasions, you need to find the registry subkey that corresponds to a fix you need to make. Other times, you need to change the name of the actual subkey (e.g., because of an organizational change). But how do you find these subkeys? And what if you need to edit not just your local machine's registry but also remote machines' registries? The answer to these questions is Regfind, which you can find in the *Microsoft Windows 2000 Server Resource Kit Supplement One*. (The tool also works with Windows 9x registries.)

To use Regfind in its simplest form, you feed the utility a string of characters to search for. For example, the command

```
regfind "acme"
```

searches the registry for any keys containing the string `acme`. You can also use Regfind to replace one string with another: Simply add the `-r` option and the new string. For example, suppose your company recently changed its name from Acme Ltd to Apex Industries and you want all the product-registration information in the registry to reflect that change. You would type

```
regfind "Acme Ltd"
-r "Apex Industries"
```

When a space or punctuation mark appears in a search string, you must enclose the string in quotation marks.

If you needed to make the same change on a remote machine's registry, you would add the `-m` option followed by the server's Universal Naming Convention (UNC) path, as in

```
regfind -m \\server03 "Acme Ltd" -r "Apex Industries"
```

To speed up Regfind, you can limit its search to a particular registry hive or subkey. Use the `-p` option to specify the registry location you want to search. For example, suppose you want to narrow your search for Acme Ltd to the `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft` subkey. To restrict Regfind's search-and-replace efforts to subkeys within that subkey, you would type

```
regfind "Acme Ltd"  
-r "Apex Industries"  
-p "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft"
```

Regfind is a helpful tool, but I've discovered a few drawbacks. First, Regfind doesn't seem to support wildcards or UNIX-style regular expressions. The only flexibility that you can build into Regfind is through the `-y` option, which makes Regfind case insensitive. For example, the command

```
regfind acme -y
```

would match `acme`, `Acme`, `ACME`, and all other uppercase-and-lowercase combinations. Oddly, Regfind's options are case insensitive—for example, you can type either `-R` or `-r`.

Second, when you use Regfind in a search-and-replace operation, the tool replaces all instances of the string. I haven't found a way to use Regfind to change only one instance of a string, except by using the marginally helpful method of restricting the search to a specific hive. My suggestion—and the Regfind Help suggestion—is to first use Regfind to find the item you're looking for, then either go ahead with the global replace or manually change the specific items you want to replace.

Third, if you need to modify numerical data (e.g., change a timeout parameter from 15 to 30), Regfind isn't the best tool for the task. If you were to instruct Regfind to change 15 to 30 in the registry, the tool would find every registry subkey with a 15 value and change that value to 30. Of course, you can restrict searches to a particular key's subkeys, but even this workaround's parameters are too broad. Most `REG_DWORD` values that you would want to manipulate contain values of 1 or 0, and more than one `REG_DWORD` value in a given key or subkey would probably be 1 or 0. Using Regfind to change one value would change them all.

## Srvany

Of all the capabilities that separate Windows NT from earlier Microsoft OSs, services are among my favorite. For example, if you run Web or mail server software as a service, you don't need to start up the Web or mail server software every time you reboot the system. Because the software runs as a service, it starts automatically. In addition, services run in the background without cluttering up your desktop or system tray, and they run when no one is logged on. Thus, your Web or mail server software will automatically restart after a power failure. Furthermore, services can run with more rights and permissions than the person operating the computer has.

However, for a program to run as a service, the developer must have built the program to do so. Many programs aren't designed to run automatically. For example, the nifty little Visual Basic (VB) program you whipped up to update content on your Web site every hour won't run unless someone is logged on to make the program work. Rebuilding a program to run as a service isn't difficult, but if you don't have a program's source code, you can't modify the application so that it can run as a service.

*The Microsoft Windows NT Server 4.0 Resource Kit's* `srvany.exe` tool offers a solution. Srvany lets you run just about any program as a service, without requiring you to rebuild the application as a service.

The idea behind Srvany is clever: Instead of rebuilding a program as a service, you run the Srvany service, and Srvany launches the program. Because Srvany won't launch all types of programs, you must experiment to determine whether the service will launch a particular program.

In addition, some applications will work as services only while you're logged on. When you log off, NT sends a command to all running programs announcing that you're logging off. Many interactive programs, such as word processors and Web browsers, pay attention to the *user is logging off* command and respond by shutting themselves down. Such programs won't run successfully as services.

Preparing a program to run as a service involves several steps. First, use the resource kit's `Instsrv` tool to install Srvany as a service. Every service has a registry key under `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services` that describes the service. For example, the DHCP Server service's key name is `DHCPserver`. Srvany lets you choose a key name for its service. This factor is important because you need to install Srvany as a service once for each program that you want to make a service. Installing Srvany under several key names lets you create placeholders for many programs that Srvany can then launch.

Suppose you have a program named `notify.exe` that you want to make a service. You'd install Srvany and give the service a key name such as `notify`. To start `Instsrv`, you'd enter

```
instsrv notify <fullpathname>\srvany.exe
```

You need to include the entire path to the location of `srvany.exe`. I've been able to make Srvany work without specifying a path if I put the `srvany.exe` file in the `\winnt` directory, but including the full pathname is a good idea.

You probably also need to change Srvany's service account. When you run a program from the desktop, the program impersonates you and has your permissions and rights. In contrast, a service doesn't run with your permissions and rights. Instead, services run with the permissions and rights of the System account (aka the `LocalSystem` account). The System account has a lot of power over the workstation or server that the account is running on but virtually no power outside that machine (i.e., over the network). Thus, you might need to let the service that you're running impersonate you. Open the Control Panel Services applet, locate the service, and click Startup. In the Log On As dialog box that opens, select Browse, and select your account and password.

Services such as DNS, DHCP, WINS, and Microsoft Internet Information Server (IIS) run regardless of whether someone is logged on because developers built the programs to do so. But modifying an application to run as a service is difficult if you can't access the application's source code. Srvany lets you run an application as a service whether or not you're logged on or have access to the application's source code. Srvany acts as a kind of envelope for almost any program (i.e., programs that don't require user input) to run in.

As I explained, installing Srvany as a service is the first step to set up a program to run as a service. In the installation process, you must create a registry key under `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services`. I used a fictitious application named `notify.exe` in my example, and I created a registry key named `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Notify`.

After you create a registry key for the program you want to run, you need to tell the system to run the program. Open a registry editor, and go to the registry key you created. Within the key, create a value entry named `Application` as a `REG_SZ` string and enter the application's fully qualified pathname. For example, I'd enter `D:\apps\notify\notify.exe` for the `Application` value in `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Notify`.

Some programs work only if you set the default directory to an arbitrary location. Thus, you need to know how to tell NT to point to such a location. In addition, some programs need NT to pass parameters to them. You need to know how to set up the registry parameters. To solve both of these problems, you must create two additional registry keys.

If you want `Srvany` to run an application in a particular default directory, create a value entry named `AppDirectory` as a `REG_SZ` string in `Srvany`'s registry key and put the directory name in that value entry. To specify a set of parameters to feed to the application, create a value entry called `AppParameters` as a `REG_SZ` string in `Srvany`'s registry key and put the parameters in that value entry. Most of these operations are case insensitive. However, if the program (e.g., `notify.exe`) requires parameters to be in a particular case, you must use the correct case when you enter those parameters in the `AppParameters` value entry. So, if `notify.exe` needed to work by default in the directory `E:\files` and required a parameter string such as `-e -i -Aupdate`, you'd have three value entries in `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Notify`: `Application`, which would contain the string `D:\apps\notify\notify.exe`; `AppDirectory`, which would contain `E:\files`; and `AppParameters`, which would contain `-e -i -Aupdate`.

After you create the necessary registry keys to let an application run as a service, you have an important decision to make. You must decide whether you want the service to interact with the console (i.e., use the local keyboard, screen, and mouse) or access network resources; NT won't easily let a service do both.

Start the Control Panel Services applet, find the newly installed service, click the service, and select `Startup`. The dialog box that opens lets you configure the service to start automatically, manually, or not at all (i.e., to disable the service). At the bottom of the dialog box, you must specify an account to associate the service with.

To make a service interactive, select the *Allow Service to Interact with Desktop* check box to let the service use the `LocalSystem` account. The program won't be able to access network resources. To let a service use network resources, associate the service with an account that has network privileges. The `LocalSystem` account is powerful, but only on the account's machine. In general, the `LocalSystem` account has no permissions on other machines.

After you make the necessary registry edits and Control Panel adjustments, your program is ready to run as a service. Remember that you can't make every application into a service, so you need to experiment to determine whether `Srvany` can make your favorite program run as a service.

## Tlist and Kill

I often notice that Windows 9x has many benefits over Windows NT. For example, Win9x lets you easily handle PC Cards and PCI cards; in fact, you can handle most hardware more easily in Win9x than you can in NT. In addition, most popular games don't run under NT, and those that do run badly. Yet, most applications that run on NT also run on Win9x. So occasionally I ask myself why I use NT.

Then I remember why. NT is far better at multitasking and is more stable than Win9x. NT gives me more control over my system. And when a process goes awry, NT lets me use Task Manager to kill the process.

In my experience, Task Manager successfully ends troubled processes more often than does Win9x, which frequently locks up the system. But sometimes even Task Manager can't stop a process, presenting instead a dialog box saying *The operation could not be completed. Access denied.* As a systems administrator, I resent the system denying me access, so when Task Manager can't stop a process, I use the *Microsoft Windows NT 4.0 Resource Kit's* Tlist and Kill utilities.

Tlist and Kill have been around since the very first resource kit, so you might be familiar with the utilities. Tlist is a command-line utility that lists all tasks running on the computer on which you carry out the command. (To show the processes running on a different computer, you need to use the Pulist utility.) Run Tlist without options to list all the processes running, as well as the process identifier (PID). You need to use the PID to sic the Kill utility on a process.

Kill ends all instances of the process you specify. In its simplest form, Kill uses the syntax kill PID.

To end a process, first run Tlist to reveal the names and PIDs of all processes running on the server. Then run Kill, specifying the PID for the process you want to end. For example, to end a locked-up process named dbserver.exe, type

```
tlist
```

and determine Dbserver's PID. Assuming the PID is 306, you then type

```
kill 306
```

to stop the process.

Occasionally, Kill reports that it can't stop a process. In that case, you can bring out the big guns and add the -f option (i.e., kill -f PID). In my experience, the -f option always ends a process.

Kill's documentation explains that instead of providing a PID, you can simply specify the application's name, such as kill notepad.exe or even kill note\*. However, Kill behaves a bit oddly when you kill a process by name rather than by PID. For example, when you specify the PID to kill a copy of Notepad that contains unsaved text, Notepad ends immediately. But if you type kill notepad.exe to kill Notepad, Notepad presents a dialog box asking you to confirm that you don't want to save the text before exiting. In this case, you can type kill -f notepad.exe to force Kill to ignore the message, and Notepad will shut down without a peep.

You can also use options with Tlist. The utility's -t option dumps process trees in addition to processes. For example, the -t option shows that the Windows Explorer process started the Microsoft Outlook process. You can also type

```
tlist <PID>
```

to list specific information about the specified process, such as how much memory it uses, which DLLs it employs, and the number of threads it has spawned.

Before Task Manager became available in NT 4.0, Tlist and Kill were the only tools you could use to end a wayward process. Sometimes the oldest tools are the best tools.

## Uptime

Uptime is a Windows 2000 Professional command-line tool that reads a local or remote server's System event log and produces reports about the system's uptime. Type Uptime on a command line to get a report about the local machine, or type a machine name after the command to produce a report about a remote server. A summary like the one that Figure 1 shows will tell you exactly how long your system has been up.

### FIGURE 1:

*Example of a simple Uptime report*

```
\\webpc2 has been up for: 50 day(s), 21 hour(s), 28 minute(s), 51 second(s)
```

If you append the /v and /s switches to the command, you'll get a verbose listing of every instance in which the server has gone down, been booted up, or suffered a blue screen. You'll also get an availability percentage, so you can wave a five-nines (99.999 percent) availability report at your boss if your system has been up long enough.

Uptime interrogates the System log and extracts the shutdown, startup, and blue screen messages (if you haven't disabled the blue screen messages in Control Panel). Uptime uses the machine's *heartbeat* to return the statistics. Introduced in Windows NT 4.0 Service Pack 4 (SP4), the heartbeat is a last-alive timestamp that Windows writes to the registry at 5-minute intervals. The OS uses the heartbeat in conjunction with the dirty shutdown event (System log event ID 6008) to estimate how long the system has been down. Although Uptime's ability to report availability statistics depends on the heartbeat, you can still use Uptime to interrogate servers whose heartbeat is disabled about how long they've been up.

## Chapter 2

# Desktop Tools

## Clear Screen Saver

Clear Screen Saver lets you keep a server's desktop visible without leaving the server insecure. Clear Screen Saver's value isn't immediately obvious, so let me explain why you'd want to use it.

Now and then, I want to be able to walk away from a computer without logging off. For example, I might want a server to continuously run Performance Monitor, and I'd like to be able to glance at the Performance Monitor screen as I walk past the machine. Or maybe I'd just like to leave a workstation running because it's doing computationally intensive work, such as rendering a 3-D image, that I'd like to keep tabs on.

I could leave the session up and running and a screen saver with password protection to guard it, but then I wouldn't be able to glance at the screen as I walked by—I'd have to move the mouse to stop the screen saver and type my password before I could see the screen. Of course, I could choose not to install a screen saver on the system so that the computer's desktop would remain visible. But that approach would require me to log on and walk away, which isn't a good idea from a security standpoint.

Therein lies the genius of Clear Screen Saver. It's a screen saver that you can password-protect, but when the screen saver kicks in, it doesn't change the screen. You see the desktop as it would appear if the screen saver weren't running. The best part is that even though you're logged on, your session is secure: If someone tries to sit down and do something at the keyboard, the screen saver demands a password.

Clear Screen Saver is on the *Microsoft Windows 2000 Server Resource Kit Supplement One* CD-ROM in the Apps\clearscreensaver folder. That folder contains just two files: Clear Screen Saver.scr and ClearSaverSvc.exe. Copy them both to C:\winnt\system32. The .exe file is a service; to install it, type

```
clearsaversvc -i
```

Or, if you've already copied that file from the resource kit CD-ROM and have installed runext.inf, simply right-click clearsaversvc.exe, choose Run, add the -i switch, and press Enter.

## Cliptray

Many *Microsoft Windows NT 4.0 Resource Kit* utilities are like automobile jacks: These tools are godsend when you need them, but—if you're lucky—you seldom use them. However, I recently found a utility that I've incorporated into my suite of tools and that I use almost every day: Cliptray. You can find Cliptray in Supplement 4 of the resource kit. (Cliptray is also available in the *Microsoft Windows 2000 Resource Kit*.)

Much of the reader mail that I get comes from people asking for help with particular problems; inevitably, I find that certain questions arise again and again. To save time, I could keep a database of answers to frequently asked questions and reuse those answers. I tried keeping a

desktop folder that contained text files of common questions and answers, but to retrieve an answer, I had to find and open the file for a particular question, then copy the text into my email reply to the reader—a lot of work. How could I simplify the process?

Cliptray is the perfect solution. This utility creates a simple database-type file that stores blocks of ASCII text; the file contains only the ASCII text blocks and a title for each block. Cliptray sits in my desktop's system tray. When I need a text block, I just right-click the Cliptray icon, and up pops a list of text block titles. I then click the desired title, and Cliptray puts that block's text onto the clipboard. With two more clicks, I paste the text into my email message.

To create new text blocks, you can use the Add option from Cliptray's context menu, or you can simply edit the text-block file—it's an ASCII file named `cliptray.txt`. To change the file's name or location, you can go to Options, File, Open, Create from Cliptray's context menu, or you can directly edit `cliptray.txt`. Each `cliptray.txt` entry includes the following elements: the word Title followed by a colon and the name of the text block (on one line, no spaces), the text block, and the word End followed by a colon (on one line, no spaces). An entry, then, might look like the following:

```
Title:createusers
You can use the NT 4.0 User Manager program, which resides in the
  Administrative Tools group, to create users.
End:
```

Cliptray doesn't require you to put `cliptray.txt` in a specific location. Tell the utility once where you want to keep the file—I keep mine on the server so that I can easily keep my canned answers with me when I switch workstations—and Cliptray remembers the location the next time you start up. Cliptray even lets you specify the file by its Universal Naming Convention (UNC) name; many other programs seem sadly incapable of this simple touch.

The primary trouble that I ran up against while working with Cliptray occurred during installation. To function, the utility needs three program files: `msgblast.ocx`, `vb40032.dll`, and `cliptray.exe`. For reasons I haven't discovered yet, Cliptray is picky about where you put those files. When I ran the utility on an NT 4.0 workstation, installation was simple—I put the `.ocx` and `.dll` files in the `\winnt\system32` directory and put `cliptray.exe` wherever I wanted, and Cliptray ran smoothly. But to make Cliptray work on a Win2K Professional workstation, I needed to put `msgblast.ocx` on the desktop. (I didn't even need `vb40032.dll`, probably because some other application on the system had already loaded the Visual Basic—VB—runtime, and I could still put `cliptray.exe` anywhere on the computer.) However, I've heard that this odd behavior doesn't occur on Win2K systems when you use the Cliptray version that comes with the Win2K resource kit.

I can imagine VBScript macro writers making good use of Cliptray, perhaps by keeping often-used code fragments in a Cliptray file. The utility can handle large text blocks: I created a seven-page block of text, and Cliptray stored it without any trouble. Sometimes small is beautiful—that is certainly the case with Cliptray. I strongly suggest that you give this tool a whirl.

## Cmdhere

Cmdhere (`cmdhere.inf`) modifies Windows Explorer so that you can right-click any folder and choose CMD Prompt Here. A command-prompt window then appears, and the default directory is already set to the folder that you right-clicked.

Why is Cmdhere useful? Because long pathnames make directory navigation a pain. For example, suppose you want to run a command-line utility from the *Microsoft Windows 2000*

*Server Resource Kit.* You need to open a command prompt, then type a command and path such as

```
cd "c:\program filesresource kit"
```

As if that process weren't inconvenient enough, you're probably already looking at C:\program files\resource kit in Windows Explorer, yet Windows Explorer still makes you do all that typing. If you install cmdhere.inf, you can simply right-click the resource kit folder and choose CMD Prompt Here.

To install cmdhere.inf, just right-click the file in Windows Explorer, then choose Install. Cmdhere.inf isn't perfect—you must right-click the actual folder icon; you can't just right-click the empty background inside a folder—but it's useful.

## Runext

Runext (runext.inf) makes running command-line routines that need parameters a snap from the GUI. Some of the best utilities are the simplest. Runext is a great example of that. Here's the scenario: You're using Windows Explorer to browse a directory in My Computer, and you want to start up an executable file in that directory. You can double-click the file to start it. That ability is convenient if you don't have a prebuilt icon for a program or can't remember its exact name.

But now consider how inconvenient the start-the-program-from-the-GUI approach is when you need to feed the program a parameter or two. For example, perhaps you're going to start up WordPad (\winnt\system32\write.exe) and want to specify a document that it should open. From a command line, you could type

```
C:\winnt\system32\write.exe  
C:\data\myfile.txt
```

But that's quite a bit of typing. Instead, with Runext, you can just right-click the write.exe file and choose Run. A dialog box then appears with a text field that already contains the C:\winnt\system32\write.exe path. You need only fill in the additional parameters—C:\data\myfile.txt, for example—and click OK to run the command.

Runext is a real time-saver if you run programs that need parameters. You can find runext.inf on the *Microsoft Windows 2000 Server Resource Kit Supplement One* CD-ROM in the Apps\RunExt folder. Simply right-click the runext.inf file and choose Install. This tool is also available in the *Microsoft Windows NT Server 4.0 Resource Kit Supplement 4*.

## TweakUI

TweakUI isn't an administrative utility, but administrators love it. Heck, everyone loves it. Shortly after Windows 95 first appeared, Microsoft placed a set of cool utilities called PowerToys on its Web site. TweakUI was probably the most popular of these utilities. It's a Control Panel applet that lets you tweak the UI. Basically, TweakUI is the result of a Microsoft programmer's efforts to give a bunch of useful registry settings a nice GUI.

After you install the *Microsoft Windows NT Server 4.0 Resource Kit* or *Microsoft Windows NT Workstation 4.0 Resource Kit* on your machine's hard disk, find the PowerToys folder. Inside PowerToys, find a file named tweakui.inf. Right-click the file, and select Install. TweakUI will install, and you'll see a Help file about the utility.

Open Control Panel, and you'll find that you now have the TweakUI applet. Open the applet, and you'll see a dialog box with tabs labeled Mouse, Network, General, New, Add/Remove, Explorer, Repair, Desktop, My Computer, and Paranoia. (No, I'm not kidding.)

### **TweakUI Tabs**

The Mouse tab controls mouse sensitivity to double-clicks and drags; this functionality is no big deal. But, it can also speed up the appearance of your Start menu. You can configure in the registry the duration of the delay between the time you click Start and the time the menu appears; TweakUI provides a nice slider to control the delay, so you can play around with the delay and discover which setting you like best.

The Network tab answers a question I often hear: "How do I get my NT machine to automatically boot and log me on without waiting for me to type a password?" If you tell TweakUI your name and password, NT will eliminate your logon sequence and immediately take you to the desktop when the system boots. (Bypassing your password obviously creates a security problem, but you might want to use this TweakUI feature on your home computer.)

The General tab provides check boxes that let you turn on and off error beeps, smooth scrolling, and window animation. You can also specify locations for your Desktop, Favorites, My Documents, Recent Documents, and other user-specific folders. This functionality is useful because NT tends to stick all of its files on the drive you install the OS on. If that drive fills up, you might be able to free up necessary space by moving the desktop to another drive.

The Explorer tab lets you control how Explorer identifies an icon as a shortcut rather than an object. You can tell Explorer not to automatically prefix a shortcut's title with *Shortcut to*, and you can suppress the small boxed arrow that appears by default in the lower left corner of shortcut icons. If you turn off your Welcome Screen tips, you can use the Explorer tab to turn them back on.

The Repair tab rebuilds icons, repairs the font folder, and repairs system files. The New tab lets you customize the menu that appears when you right-click a container and select New. For example, on my system, I can right-click my C drive and create a new text file, HTML file, or Microsoft Word document. The New tab lets you add file types to the menu's list and remove file types from the list.

Paranoia has two subsections: Covering Your Tracks and Things That Happen Behind Your Back. The first section lets you tell your system to automatically erase the queue of recently visited documents when you log on so other people can't easily see which files you've been using.

The second section lets you stop music CDs and Autorun CD-ROMs from automatically playing when you put them in the CD-ROM drive. Removing that functionality is one of the first things I do when I set up a new system; getting a *Do you want to install NT?* message every time I place the NT Server CD-ROM in my CD-ROM drive is incredibly irritating.

### **Winexit**

You adopted Windows NT because it's supposed to be a secure OS. You use ACLs to secure most objects on your NT network, and you prevent users from accessing NT workstations without a password. You implemented Service Pack 3's (SP3's) cool *passfilt.dll*, which forces users to choose complex, difficult-to-crack passwords. Running NT on desktops throughout your enterprise seems like a great way to keep your network secure, right?

Your network might not be as secure as you think it is. Many networks let users stay logged on indefinitely. If you walk around many corporations, you'll see NT desktops that users have logged on to and walked away from. Unattended desktops weren't a problem when networks ran off mainframes because mainframes automatically log off users after a certain period of inactivity. How can you perform such an automatic logoff in NT?

You can use Winexit to secure inactive workstations. This screen saver program ships in the *Microsoft Windows NT Server 4.0 Resource Kit*. Winexit consists of one file, winexit.scr, which you can find in the resource kit directory.

## Winexit Options

Right-click winexit.scr and you'll see the options Install, Test, and Configure. Select Install. A Display Properties dialog box will appear. The Display Properties dialog box shows the Screen Saver tab from the standard Control Panel Display applet; the Screen Saver dropdown menu will have the Logoff Screen Saver option selected.

You can change the value in the Wait spin box to select how long you want your network's computers to wait from the time users become inactive until Winexit starts the logoff process. The default Wait value is 15 minutes.

After the Wait period expires, Winexit starts. The utility displays an *Auto Logoff in progress* dialog box that warns users that Winexit is going to log them off. Users can click Cancel or press any key to stop the logoff process. The dialog box counts down for a period of time (30 seconds by default). When the period expires, Winexit logs off the user.

To change the length of time the *Auto Logoff in progress* dialog box counts down, click Settings on the Screen Saver tab. You can configure three settings in the Winexit Setup Dialog box that appears: Force logoff, Time to logoff, and Logoff Message. You configure the logoff countdown period in the *Time to logoff* section's Countdown text box. The text box's value is the length of the logoff countdown in seconds. Winexit accepts values from 0 to 999. If you set the value to 0, the computer will wait for the period you specify in the Wait spin box, then log off users without giving them a chance to avert the logoff.

The Logoff Message text box lets you customize the *Auto Logoff in progress* dialog box. Double-click the Winexit icon to see the *Auto Logoff in progress* dialog box; the message you enter in the Logoff Message text box replaces the default message *Use Setup to change the text in this line*. You can leave the Logoff Message text box empty or enter a message such as *The network is going to log you off because your machine is inactive* or *To maximize network throughput, the network automatically logs off inactive sessions*.

The Winexit Setup Dialog box's *Force application termination* check box lets Winexit terminate users' applications without saving their data. When users log off NT workstations, they receive messages from applications that have open, unsaved files. These dialog boxes question whether users want to save unsaved data. The default Winexit logoff process waits for users to respond to applications' dialog boxes before logging the users off. However, users who aren't at their desk can't choose to save or reject changes to documents.

If you don't select the *Force application termination* check box, Winexit won't log off users who have unsaved data. If you select the check box, Winexit won't wait for users to respond to applications' logoff dialog boxes, and users will lose unsaved data. Whether you need to select the

## 18 A Guide to Windows Power Tools

*Force application termination* check box depends on your company's policies and whether all your users diligently run their software's automatic save options.

Regardless of whether you choose to terminate programs that have unsaved data, you can use Winexit to make your network more secure. Make Winexit your next system policy.

## Chapter 3

# Diagnostic Tools

## Browser Monitor and Domain Monitor

Windows NT networks contain machines that serve many functions. Two of the most important NT functions are browse master and domain controller (DC). You usually don't need a utility to discover which of your machines are DCs because you must explicitly create your DCs. But computers become browse masters because the other NT and non-NT machines in their workgroup elect them to that task, and sometimes you need to figure out which computer is acting as browse master. The *Microsoft Windows NT Server 4.0 Resource Kit* includes a tool for finding your workgroup's browse master and a similar tool for finding DCs.

### *Jewel of a Tool*

Imagine that your shop has stayed with NT 3.51 servers and you run Windows 95 on users' desktop systems. People start complaining that the Network Neighborhood is randomly misbehaving. You open Event Viewer and see that your DC is refusing to act as browse master. The DC claims that another computer has asserted that it is the browse master. How can you find out which computer is causing your problems?

Use the resource kit's Browser Monitor to determine which machine is acting as browse master. Invoke the tool from the command line by typing

```
browmon
```

or start it from the Programs menu by selecting Start, Programs, Resource Kit 4.0, Diagnostics, Browser Monitor.

After you start Browser Monitor, select Domain, Add Domain and enter the name of the workgroup whose browsers you want to find.

In this scenario, you'll probably find that a Win95 workstation has seized browser mastership from your DC. If so, the Win95 machine will appear in Browse Monitor. How can you solve this problem? Simple. Either turn off your Win95 machines' file- and print-sharing module or set `MaintainServerList` to `No` on the Win95 machines. (You can use a system policy to set `MaintainServerList` to `No`, or you can make the change through the Control Panel by selecting Networking, File and Printer Sharing, Properties, Advanced.)

### *Monitor Mystery*

Domain Monitor, a tool that sniffs out DCs, appears near Browser Monitor on your Programs menu, but Domain Monitor doesn't work correctly. If you don't believe me, try the following test: Install NT on a server and make that server the PDC of a new domain. Install the resource kit (or just `dommon.exe`) on your new PDC. Then, run Domain Monitor and ask it which machine is your new domain's PDC. Domain Monitor will probably give you the message *PDC not found*.

Why does Domain Monitor have this quirk? I don't know. Microsoft includes many utilities in the resource kit rather than the shrink-wrapped OS because resource kit utilities don't require as much quality testing. The resource kit includes some stellar tools, but it also includes tools that just don't work. For example, Microsoft claimed that a basic Internet server tool that used to be in the NT Server 4.0 resource kit offered SMTP and POP3 services, but I have never been able to make the tool work.

Microsoft claims it doesn't need to support resource kit tools because it doesn't sell the tools; it sells the resource kit book, and the tools come free as a bonus. This argument seems kind of odd in light of the fact that the *Microsoft Windows NT Server 4.0 Resource Kit Supplement Two* is nothing more than a revised set of resource kit tools on two CD-ROMs, and Microsoft certainly charges for Supplement Two.

Because Microsoft doesn't support resource kit tools, the tools' users must take the bad with the good. Use Browser Monitor, but leave Domain Monitor alone.

## Driver Verifier

Your system's been producing blue-screens, and you're trying to figure out what the cause is. You installed that new service pack, and the notes said something about an improved network redirector ... or could that updated NIC driver be causing the problem?

Driver Verifier is a utility that is free but that doesn't come in a resource kit. The utility is in Windows 2000, in both the Professional and Server incarnations. Despite its name, Driver Verifier isn't just a driver verifier—it tells Win2K to closely monitor and verify any kernel-mode module's behavior on your system. In Driver Verifier, Microsoft has given us a tool that in effect lets us do a quality-assurance check on any or all of the system's low-level parts.

To start the Verifier, click Start, Run, then type Verifier and press Enter to view the Driver Verifier Manager window. Click the Settings tab, and you see a *Drivers currently active in the system* pane that displays a list of eligible drivers. The listed drivers include some, such as classpnp.sys (Plug and Play—PnP—support), efs.sys (Encrypting File System—EFS), and fastfat.sys (32-bit FAT support), that are more than hardware drivers, though—they're essential parts of the OS.

Stated very simply, all Win2K and Windows NT pieces are either kernel mode or user mode. *Kernel-mode* modules are generally more powerful (which makes them attractive to developers) but also more capable of accidentally damaging another kernel-mode program's memory areas. Such damage tends to make the program crash and trigger a blue screen, and determining which module committed the damage is difficult. The OS watches *user-mode* modules more closely than kernel modules: When user-mode modules try to write out of their areas, the OS stops them and wakes up Dr. Watson to tell you.

The beauty of the Verifier is that it basically tells the OS to watch a given driver or group of kernel-mode pieces far more closely, almost as if they were user-mode pieces. Point the Verifier at a module, and the Verifier will watch how the module calls the OS and will check for nonsensical or illegal memory requests and API calls. The Verifier also will examine the module's memory upon driver unload, looking for leftover but unresolved pending actions, such as unfinished I/O operations.

Select the *Special pool* check box in the *Verification type* pane, and the OS will allocate the driver's memory in a location that is surrounded by unused memory areas that are marked "nobody reads, nobody writes." So, if the driver strays out of its assigned memory area, Win2K will

immediately detect the problem and force a blue screen, and the incriminating evidence will point straight at the bad kernel piece.

If you select *Pool tracking*, Verifier will watch for memory that the kernel module allocated but never deallocated, a dreaded condition known as *memory leak*. Pool tracking smokes out the leaking module and forces a blue screen. (Too bad this option doesn't work for user-mode applications, some of which are among the worst leakers.) I also recommend selecting *Force IRQL checking*, which monitors for behaviors in which an application might try to access pageable memory areas (i.e., areas marked to be moved to the pagefile and that thus are no longer trustworthy), another source of mysterious errors. You probably shouldn't select the *Low resources simulation* option, which randomly reports out-of-memory conditions to the kernel application. The option is a good test, but it really slows down the system, and well-built kernel applications should be able to handle low-memory conditions. However, you should select I/O verification and set it to level 2. This setting specifically identifies I/O violations; for example, the blue screen might say *BugCheck 0xC9 (DRIVER\_VERIFIER\_IOMANAGER\_VIOLATION)*. Finally, after you enter or change Verifier settings, you need to reboot.

Driver Verifier comes with a price: It will slow down your system a bit, and the more drivers you watch, the slower the system runs. But Verifier's ability to identify a problematic kernel module lets you check out a new server or new system software before you install it on the network. Isolate the new server or software—quarantine it, if you will—and run Verifier for a few days. Its deliberate blue screens might save you from unexpected ones later.

## OH

OH (for “open handles”) helps when you try to delete a directory only to have the OS tell you that a file in the directory is “in use by another process.” In the past, I've used Sysinternals' great Filemon utility (available at <http://www.sysinternals.com>) or an old copy of Windows File Manager from Windows NT 3.51 to determine which process was using the file. But I recently discovered a *Microsoft Windows 2000 Resource Kit* tool called OH that does the job. (To my embarrassment, OH apparently has been around for quite a while.) OH has many options. I'll give you the short version for finding out what program is preventing you from deleting a file.

Before OH will work, you need to run it once. The first invocation modifies Win2K's kernel a bit, adding 8 bytes of information to each running process. Theoretically, that overhead will slow the system, but I've done before-and-after comparisons on several systems and have found no detectable difference in performance or memory utilization. (The resource kit Help file says that you can undo the kernel change later with a tool called *gflags.exe*, but I haven't tried it.) The worst part of using OH seems to be that a reboot is required for the kernel behavior modification.

After rebooting, you can tell OH to identify the process that's using any given file by typing the command

```
oh -t file <filename>
```

The *-t* switch tells OH that you want information about a particular type of object—a file, in this case. *Filename* is the name of the file you're interested in.

If you're unsure of the file's name, OH can still help. You can't use the common wildcards (i.e., *?* and *\**) with OH, but you can type the first few characters of the file's name to have OH list

## 22 A Guide to Windows Power Tools

all the files in use whose names begin with those characters. Thus, if you know that the file's name is process1.dat or process2.dat, you can just type

```
oh -t file process
```

to see whether a file called process1 or process2 is in use.

If you know only the file's extension, type

```
oh -t file
```

to dump the name of every file in use. Then, pipe the output through the Find filter to narrow the list to a particular extension. For example, to find all files in use that have the extension .pl, type

```
oh -t file | find ".pl"
```

## Chapter 4

# File and Disk Tools

## Compress

Many Microsoft products come packaged as a directory of files compressed with a Microsoft compression algorithm. You can recognize a compressed file by the final character in its file extension: an underscore (e.g., after compression, test.txt becomes test.tx\_).

You'll occasionally want to replace files from the Windows NT installation CD-ROM's i386 directory with new and improved files, and that's when the ability to perform your own file compression can come in handy. For example, suppose your hardware desperately needs driver xyz.sys, but the distribution version, xyz.sy\_, is buggy. You do installs from a central, networked i386 directory, and you want to make future installs go smoothly. You get an improved xyz.sys file, and you want to ensure that future NT installs work from the improved driver rather than the original one. According to Microsoft documentation, all you need to do is simply copy the new xyz.sys file to the i386 directory (i.e., you don't need to compress the file). If NT Setup finds both an xyz.sys and an xyz.sy\_, Setup will use the uncompressed version.

Unfortunately, that rule doesn't always work. Setup zeroes in on the compressed versions of some device drivers. To be safe, I use *the Microsoft Windows NT Server 4.0 Resource Kit's* Compress tool (compress.exe) to revise files before I put them into my distribution i386 directory:

```
compress -r xyz.sys
```

The -r option tells Compress to create xyz.sy\_, the compressed version of xyz.sys. Compress creates the compressed file in the same directory as the uncompressed file. Alternatively, you can specify a directory for Compress to put the compressed files in and crunch up an entire directory:

```
compress -r *.* c:\crunched
```

## Fileman.vbs

Now and then I need to change ownership of a file that is on an NTFS drive. I can do that with the GUI through My Computer or Windows Explorer (select the file, right-click it, choose Properties from the resulting context menu, choose the Security tab, and click the Ownership button), but as is so often the case, I'd like to be able to do this task from the command line as well.

Windows NT resource kits have long included the Chown utility for changing ownership, but I've never been able to make it work. So, I was excited to find fileman.vbs tucked among the dozens of VBScript routines in the *Microsoft Windows NT Server 4.0 Resource Kit Supplement 4*.

Fileman.vbs lets you copy, delete, rename, or seize ownership of a file. Command-line tools already exist to copy, delete, and rename files, but I was glad to finally have a command-line ownership tool. The tool is a mite cumbersome, as you'll see. But because it's a script file, you've got the source code, so enterprising readers can address some of fileman.vbs' flaws.

To change a file's ownership, you invoke `fileman.vbs`:

```
cscript <scriptpath>\fileman.vbs /T <fullyqualifiedfilename>
```

The syntax looks straightforward, but kinks exist. First, notice the `scriptpath` part—you feed the name of the VBScript file to `cscript`, so you need to provide `fileman.vbs`' qualified name and location. Because my resource kit files are in `C:\ntreskit`, my command would begin with `cscript c:\ntreskit\fileman.vbs`. Second, the `/T` option is case sensitive. Third, you must specify the fully qualified name of the file that you want to seize ownership of. Typically, when you execute a command from within the same drive and directory as the file on which the command acts, you can omit the file's directory and drive letter. Not here! To take ownership of the `names.txt` file on `D:\myfiles`, I'd need to type

```
cscript c:\ntreskit\fileman.vbs /T d:\myfiles\names.txt
```

Finally, you can't feed `fileman.vbs` wildcards—you can't simply tell `fileman.vbs` to change ownership on `*.*` and walk away secure in the knowledge that `fileman.vbs` will take ownership of all the files in a directory. You can work around that limitation, however, with a fairly long `For` statement.

For those who've not used it, the `For` command lets you tell NT to repeat a specified command for a group of files. Simplified, the `For` command looks like

```
for %f in (*.*) do <somecommand> %f
```

This command causes NT to find all files that meet the criterion in the parentheses—I used `*.*`, but you could use anything—and to replace `%f` with each file's name, one at a time. With each replacement, the `For` command executes the specified command, again replacing `%f` with the current filename. For example, to tell `fileman.vbs` to take ownership of every file in a directory named `C:\soontobemine`, I'd type

```
for %f in (c:\soontobemine\*.*) do cscript c:\ntreskit\fileman.vbs /T %f
```

Because I specify a drive and directory name in the parentheses, each execution of the specified command includes that drive and directory name, so I don't need to type the drive and directory in the right-hand part of the command. If I wanted to execute the `For` command from inside `C:\soontobemine`, then I could have a shorter term in parentheses, but I'd need a longer right-hand portion:

```
for %f in (*.*) do cscript c:\ntreskit\fileman.vbs /T c:\soontobemine\%f
```

One final note: If you want to include a `For` command in a batch file, you must change `%f` to `%%f`, as in

```
for %%f in (c:\soontobemine\*.*) do cscript c:\ntreskit\fileman.vbs /T %%f
```

## Freedisk

Occasionally, a problem with hard disk space crops up when I'm writing batch files that set up directories. For example, when I copy the `i386` directory from a central distribution point to a local hard disk, sometimes the hard disk doesn't have enough free space. When I set up my Windows

NT seminars, I take a laptop with the NT source files and ask the client to provide a demonstration machine that I can install NT onto. One of the first things I do is copy the i386 directory from the laptop to the demonstration machine's local hard disk. Because NT's command interpreter integrates better with the network redirector than DOS's command interpreter did, copying a set of files from a share named i386 on a server named Source is simple. I just type

```
xcopy \\source\i386 c:\i386 /s
```

The process doesn't require the mapping of drive letters.

Once in a while, the demo machine doesn't have enough free space on its hard disk. I don't always check for free space, but when I do, the *Microsoft Windows NT Server 4.0 Resource Kit's* Freedisk tool helps out. Here's the syntax for Freedisk:

```
freedisk driveletter required_number_of_bytes
```

Freedisk returns OK if the drive has the requested number of bytes (not kilobytes or megabytes), or *To small!!!* (the resource kit output proofreader slept through this one) otherwise. Freedisk also sets a return code of 0 to indicate sufficient space or 1 to indicate insufficient space. You can eliminate the unnecessary OK or To small!!! notices with a redirection, as I've done in the following example:

```
@echo off
freedisk c: 300000000 >NUL:
if errorlevel == 1 goto nogood
xcopy \\source\i386 c:\i386 /s
goto quit
:nogood
echo Not enough space to copy i386 files and do an NT installation
:quit
```

## Permcopy and Share.vbs

Suppose it's migration time and you're rolling out Windows 2000. You've got a lot of work to do: Design the DNS and Active Directory (AD) structures, migrate user accounts, upgrade some servers, and scrap and replace some others. If some of those new servers are file servers, you face an old problem: copying a bunch of files and folders from a share on one machine to a share on another machine while minimizing the amount of work you need to do to recreate the file, directory, and share permissions on those file shares. Permcopy and share.vbs are two tools that can help you get started.

Permcopy.exe duplicates the share permissions that it finds on one share on another share. The syntax is

```
permcopy \\sourceserver sourceshare \\targetserver targetshare
```

This command duplicates the \\sourceserver\sourceshare share-level permissions on \\targetserver\targetshare.

Permcopy has four arguments: the names of the source server, source share, target server, and target share. The command's syntax is a bit nonintuitive. Rather than using Universal Naming Convention (UNC) names for the shares, you separate each argument by at least one space from the

rest of the command string. For example, to duplicate the permissions on the Mystuff share, which is on a server named Homebase, to the Yourstuff share on a server named Foreign, I wouldn't enter

```
permcopy \\homebase\mystuff \\foreign\yourstuff
```

Instead, I'd type

```
permcopy \\homebase mystuff \\foreign yourstuff
```

You need to know a few things about Permcopy. First, despite what the *Microsoft Windows 2000 Resource Kit* documentation says, Permcopy doesn't make the file permissions on the target directory match the file permissions on the source directory; the utility duplicates only the share permissions. Second, Permcopy doesn't copy the files from one share to the other, nor does it create shares in the target directory.

You might think, "All right, then; how will I create that new share?" You could use the GUI to create the share, of course, but let's assume that you want to perform the task on the command line and keep it as scriptable as possible. Share.vbs can list, create, and delete shares on a remote system. (Share.vbs can't set share permissions on a share; you use Permcopy to set the permissions.)

Simplified, the share.vbs syntax is

```
cscript share.vbs /c /n <sharename> /s <servername> /p <path> /t disk
```

For example, to create a share named Yourstuff on a server named Foreign, where Yourstuff lives in the directory C:\shares\yourstuff, you'd type

```
cscript share.vbs /c /n yourstuff /s foreign /p c:\shares\yourstuff /t disk
```

As with other resource kit scripts, you can add the /u option to specify a username and the /w option to add a password. The /v option lets you add a comment to the share. To extend this example, suppose I had an administrative account called Czar on the Foreign server and a password of ivan. To add the user account information and a comment to the command in the above example, I could type

```
cscript share.vbs /c /n yourstuff /s foreign /p c:\shares\yourstuff /t disk /v
  "My first share.vbs-created share" /u czar /w ivan
```

## Xcopy

Remotely migrating a shared folder from a Windows NT 4.0 server to a Windows 2000 server requires several steps. First, you need to create the new directory on the Win2K server, share that directory, and set the share permissions. Creating the directory remotely is simple—the Md command works as well on Universal Naming Convention (UNC) names as it does on drive letters. The *Microsoft Windows 2000 Resource Kit's* Permcopy tool lets you set one shared directory's permissions to match another shared directory's permissions, and share.vbs lets you share a directory remotely.

Now, we need to finish the job by copying all those files and keeping their owner information and permissions intact. This problem isn't new; almost every NT administrator has faced it. And I believe that in NT, the answer was always the same—use the *Microsoft Windows NT Server 4.0 Resource Kit's* Scopy tool. Scopy works like Xcopy, but Scopy copies owner and ACL information as well as files and directory structures. So I was surprised to find that Scopy is no longer included in the Win2K resource kit. Instead, Win2K's Xcopy command incorporates Scopy's functionality through two new options: /O and /X. The /O option copies both owner- and file-permission information and includes that information with the file. The /X option, which works only in conjunction with /O, also transfers audit settings. So, if you've been auditing the activity on a particular file on your NT 4.0 server, you can continue to audit it on your Win2K server.

Let's wrap up our migration task with an example. Suppose my NT 4.0 file server is named \\From and its file share is named Files. I'm going to migrate Files to a share named Files on a server named \\To. To keep things interesting, let's assume that I'm not sitting at either \\From or \\To, and to keep the sample commands short, let's assume each share is on the corresponding machine's C drive.

The first step is to log on to both servers. The method I like best is the old Net Use \\server-name\IPC\$ /user:domainname\username trick, but you can use whatever method you like.

Next, create the directory on \\To by typing

```
net use X: \\to\c$
md X:\Files
```

at the command line. Then, tell \\To to share the C:\Files directory as \Files:

```
cscript share.vbs /c /n Files /s To /p C:\Files /t disk
```

Again, note the oddities about share.vbs—you refer to the To server as To, not as \\To. Next, use Permcopy to ensure that the permissions on the \\To\Files share are the same as those on the older \\From\Files share:

```
permcopy \\From Files \\To Files
```

Remember to put at least one space between \\From and Files and between \\To and Files. Finally, use Xcopy to move the files. If you want the owner information to migrate properly, Xcopy must see these moves as being from directory to directory rather than from share to share. So, we need to map to the C drive on \\From as well:

```
net use Y: \\From\C$
xcopy Y:\Files\* X:\Files /S /O /T
```

With that, we're finished. And now that I look at what we've done, I realize that I could have saved a step by going directly to Xcopy and using it instead of Net Use to create the target directory. That's the thing about Win2K: It gives you at least 20 ways to do everything.

## Chapter 5

# Network Management Tools

## Addusers

Systems administrators who need to create or delete hundreds or thousands of Windows NT user accounts en masse must appreciate `addusers.exe`, a command-line tool that comes in the *Microsoft Windows NT Server 4.0 Resource Kit*. If you have a database of users whom you want to add or delete accounts for, Addusers provides an automated alternative to User Manager. You control Addusers through simple, comma-delimited ASCII text files.

### Creating an Account

To create accounts, you must first create an ASCII text file that lists the users you want to add. The first line of the file must contain the word *user*, in brackets. The file's following lines must each contain information about one user whom you want to create an account for. The user lines need to list the username, user full name, password, description, home drive letter, home drive Universal Naming Convention (UNC), profile location, and logon batch script, in that order.

For example, suppose I want to create an account for a user whose username is `marty`, full name is `Martin Jones`, password is `opensesame`, and description is `writer`. Further, suppose Marty connects his local H drive to `\\server1\users\marty`, which he uses as a home directory; he stores his profile in `\\server1\users\marty\profile`; and he runs `start.bat` for his logon. I create an ASCII file for Marty and call it `add1.txt`. It looks like

```
[user]
marty,Martin Jones,opensesame,Writer,H:,\server1\users\marty,\server1\users\
martyprofile,start.bat
```

In the ASCII file, each user's information must appear on one continuous line; press Enter only to begin entering information about a new user. In addition, make sure that you don't add spaces after the commas. If your file includes a space between a comma and a piece of user information, Addusers will include the space in the user information.

After I create the necessary ASCII file, I must tell it to execute. Addusers' `/c` option creates user accounts, so I open a command prompt and type

```
addusers /c add1.txt
```

### Addusers' Options

You can give a user a local path instead of a home directory; just leave the home drive letter field empty and specify the local path in the home drive UNC field. For example, if I want Marty's default path to be his local C drive, I make his user line

```
marty,Martin Jones,opensesame,Writer,C:\,\server1\users\marty\profile,start.bat
```

You can also create groups with Addusers. To create a global group, add a section to your ASCII file named `global` and a line for each global group you want to create. Each global group line must contain the global group name, a comment about the global group, and the username for each user you want to include in the group. (The resource kit documentation says you must end each line with a comma, but I've found this comma is not necessary.) I can create a global group called `WriterGuys` that includes accounts for users `mark` and `marty` and the comment `Guys Who Write` by adding the following lines to my ASCII file:

```
[global]
WriterGuys,Guys Who Write,mark,marty
```

You create local groups the same way you create global groups, except that you name the section `local`, not `global`.

Finally, you can use Addusers' `/e` option to delete user accounts. Create an ASCII text file with the word *user* (in brackets) on the first line, and list the usernames of users you want to delete on the following lines. Place one username on each line, and follow each username with seven commas. When you've created your ASCII file (e.g., `kill.txt`), tell Addusers to delete the users by opening a command prompt and typing

```
addusers /e kill.txt
```

You can't control every aspect of user accounts through Addusers: You can't specify logon hours, an account expiration date, or restrictions on which workstations a user can access. You can't use Addusers to change user information in existing user accounts and groups, either. If you include the username or group name for an existing user or group in an Addusers ASCII file, you'll get an error message. Nevertheless, Addusers is a time-saving alternative to NT's User Manager that makes bulk operations easier to perform.

## Browstat

The Browser network service, which you see through its Network Neighborhood interface, must have seemed a good idea at one time, but sometimes its unreliability makes Browser seem more like a bug than a feature. Whenever Browser gives a user fits, the network administrator needs to try to figure out why it's misbehaving. But what tools does an administrator have to monitor Browser's behavior? In a word, the answer is `Browstat`.

`Browstat`, a command-line tool, tells you a lot about a network's browsers. For example, it can tell you which machine is a workgroup's browse master, list all servers that are candidates to be the browse master, and list browsing statistics for the workgroup. `Browstat`'s one annoying feature is that it requires that you do some typing to identify transport protocols.

As you might know, Microsoft network OSs elect a separate browse master for each workgroup and network transport protocol on a subnet. To find out which system is the browse master for a workgroup named `Fishes` on TCP/IP, you could type the command

```
browstat getmaster netbt_el90x1 fishes
```

The `getmaster` option tells `Browstat` to report the browse master for the `Fishes` workgroup. (`Browstat` commands aren't case sensitive.) The purpose of the `netbt_el90x1` part of the command string isn't as obvious. This part of the command is Windows NT-ese for "the NetBIOS over TCP/IP

(NetBT) stack running on your 3Com Etherlink 3C905B-TX NIC.” You must type the name correctly; `Browstat` won’t help you if you don’t. Fortunately, you can use the `Net config rdr` command to easily find the name of your system’s transport protocols. On an NT 4.0 system, the transport name will resemble `\Device\NetBT_El90x1` (you can drop the `\Device\` part). On a Windows 2000 system, the device names look like `\Device\NetBT_Tcpip_{418786EA-FAFE-443C-A9Fb-DC0CE26D87E5}`, probably because of Win2K’s Plug and Play (PnP) nature. After you’ve typed the transport name a few times, you’ll wonder, as I did, why `Browstat`’s author didn’t either include wildcard support for transport names or drop the need to type the transport for single-protocol environments.

`Browstat` has a number of options.

```
browstat status -v <workgroup-name>
```

reports the name of the master and backup browsers and lists the number and type of servers in the workgroup. The workgroup name is optional; if you don’t include it, `Browstat` reports on your workgroup.

```
browstat view <transport-name> <workgroup-name>
```

shows all servers that are potential browse masters as well as the OS that they claim to run (Samba servers commonly fib about their OS version to ensure that they’re browse masters). This command also reports the characteristics Browser uses to grade servers’ fitness to act as a browse master during an election. Again, the workgroup name is optional. For example,

```
browstat view netbtel90x1
```

will list the potential servers running TCP/IP on my workstation’s workgroup.

```
browstat getpdc <transport-name> <domain-name>
```

returns the name of the PDC for the specified domain (the domain name is optional).

```
browstat getmaster <transport-name> <workgroup-name>
```

reports the browse master for the specified transport and workgroup—information that you can also get from the `browstat stats` command.

```
browstat stats \\<systemname>
```

asks the specified system (or your local machine, if you don’t specify a system) to report its Browser statistics (e.g., how many elections the system has seen, how many times a server has announced itself).

`Browstat` is small (42KB), requires no installation, and provides a quick-and-dirty look at browsing on a network. Add it to your toolkit; `Browstat` earns its keep.

## Cusrmgr

I found `Cusrmgr` (`cusrmgr.exe`) when one of my clients was having trouble getting network administrators to be accountable for their actions. Six people in the client’s firm do all network-related administration, and everyone who needs to do administration-related tasks is a member of Domain

Admins (that's not the greatest strategy, but the client likes it). Several administrators are in the habit of using the default Administrator account to log on and do their work. That troubled the folks in charge because they wanted to be able to track what people did, but when several people use the Administrator account, not much tracking is possible.

I suggested randomizing the Administrator password, establishing a policy that no one is to change the password, and auditing password changes. Then, if an administrator changes the Administrator password, that action will show up in the log and the administrator can be reprimanded, banished, or whatever. Occasionally, you need to use the Administrator account, of course, but for those occasions, human resources (HR) can simply work out a procedure for approving its use.

Cusrmgr.exe is a file from the *Microsoft Windows 2000 Server Resource Kit* directory that lets you randomize a password. The tool's syntax is

```
cusrmgr -u <username> -p -m <machinename>
```

where username is the user account name and machinename is the name of the machine that holds the user account. (If the user account is a domain account, use the machine name of a domain controller—DC.) Specifying -p makes the password random.

Suppose I'm at a workstation in a domain named MYDOMAIN and I want to randomize the domain's Administrator password. Assuming that MYDOMAIN has a DC named DC4, I would type

```
cusrmgr -u Administrator -p -m \\DC4
```

Be sure the p is lowercase. A command with a capital P, such as

```
cusrmgr -u Administrator -P bigsecret -m \\DC4
```

sets a particular password—in this case, the command sets the Administrator account's password to *bigsecret*.

## Dhccpmd

About once a month, I get a letter from a reader looking for a program that can generate an ASCII text file showing all of a system's existing DHCP leases. Fortunately, a tool in the *Microsoft Windows NT Server 4.0 Resource Kit* offers a solution to this common problem.

### ***The Problem***

The DHCP service simplifies assigning IP addresses to computers on a network. After you designate a range of IP addresses (a *scope* in DHCP terminology) on a Windows NT server that acts as a DHCP server, you no longer need to manually enter IP addresses on your NT or other Windows workstations. Like magic, your PCs find the DHCP server, and the DHCP server gives them the next available IP address automatically.

Because DHCP gives the addresses only for a fixed length of time, the PCs lease their IP addresses from the server. The DHCP Manager that comes with NT Server lets you see a list of all currently-leased IP addresses, or *active leases*, but you can view them only through the DHCP Manager GUI.

You cannot use the DHCP Manager to output a list of active leases to a file. However, you might want to dump lease information to an ASCII file for several reasons: You need to feed information about active leases into your DNS servers (the link between WINS and DNS can be troublesome); you want to use the information as input into a homegrown network management tool; you want to back up the data; or you're just curious.

### **Problem Solved**

Dhccpmd solves the text-dump problem. Dhccpmd is a command-line tool that lists active leases, creates new scopes, and modifies DHCP server parameters. I'll focus on the active lease listing function, because it's the most useful. The syntax looks like

```
dhccpmd <serveripaddress> enumclients <scopeaddress> [-v] [-h]
```

*Serveripaddress* is the IP address of the DHCP server. You must use the server's IP address or a NetBIOS name (such as \\JUBJUB); Dhccpmd cannot resolve DNS names. *Scopeaddress* is the network number of the DHCP scope. Look in the left pane of the DHCP Manager, and you'll see the network number under the server's name. The DHCP Manager lets you use labels, such as basic scope, as the scope address, but Dhccpmd takes only the network number. If you use the -v option, Dhccpmd will give you a lot of information through long, descriptive messages. If you use the -h option, Dhccpmd will include hardware information about the machines with active leases.

Suppose you have a DHCP server named Numberman at IP address 200.200.100.7 that hands out leases from 200.200.100.10 through 200.200.100.67, and the network number of this scope is 200.200.100.0. You can ask the server to show you its active leases with either of the following commands:

```
dhccpmd 200.200.100.7 enumclients 200.200.100.0
```

or

```
dhccpmd \\numberman enumclients 200.200.100.0
```

DHCP's response might look like

```
1200.200.100.10MYIBM01/30/1998 02:32:41
2200.200.100.11D00DAD01/30/1998 02:47:41
Command successfully completed.
```

In this example, only two computers have IP addresses on Numberman, but Dhccpmd will dump IP addresses for as many computers as you have. You can redirect the output to capture the list of computers for future reference. To save the previous list, I type

```
dhccpmd \\numberman enumclients 200.200.100.0 >saveit.txt
```

Dhccpmd will save the output to saveit.txt.

Dhccpmd is a cool tool that you're likely to find useful. Just remember that you must refer to DHCP servers by their IP address or NetBIOS name (not their DNS name), and you must refer to scopes by their network number (not their scope name). In addition, I found a limitation of Dhccpmd: I couldn't run the tool on a DHCP server from a workstation in a nontrusted domain,

even though NT had authenticated me on the DHCP server with an account that the DHCP server recognized as an administrative account. Thus, a final rule: Run Dhcpcmd from NT machines in trusted domains.

## FAZAM 2000 RFV

The November 2000 TechNet shipment included a CD-ROM that contained the *Microsoft Windows 2000 Server Resource Kit Supplement One*. The original Win2K Server resource kit was terrific, except for that poisonous buy-a-copy-for-each-person license agreement. But Supplement One contains some even better tools, including a much-needed tool for scripting DNS administration. (Unfortunately, Microsoft hasn't made the software license any more palatable.)

Perhaps the most-awaited tool in Supplement One is FullArmor's FAZAM 2000, Reduced Functionality Version (FAZAM 2000 RFV), a Resultant Set of Policies (RSoP) modeling tool. FAZAM 2000 RFV is a nice—although not perfect—lite version of the commercial tool. For those who haven't used Group Policy, let me provide a bit of an introduction before I discuss FAZAM 2000 RFV.

Group Policy under Win2K is like Windows NT 4.0 system policies but considerably enhanced. Group Policy lets you control from a central location characteristics such as how user desktops behave, security policies, and software deployment. Your enterprise can have many Group Policies. When users log on, their workstations gather policies associated with the site, domain, and organizational unit (OU). Policies can override and nullify the effects of other policies, and a feature called *policy filtering* can negate a policy's effects. Thus, when users call the Help desk and ask why their computer is behaving oddly upon logon, answering the question can be difficult. Which policies actually take effect in a given situation? FAZAM 2000 RFV can answer that question.

You must install `fazam2000rfv.msi` from the Supplement One CD-ROM's `\W2Ksupp1\apps\fazam2000` directory. Then, click Start, Programs, FAZAM 2000 RFV to start the Microsoft Management Console (MMC) FAZAM 2000 RFV snap-in. You'll see two objects in the left-hand pane: FAZAM 2000 RFV Administrator and FAZAM 2000 RFV Policy Analysis.

To begin an RSoP analysis, click the plus (+) sign next to the Policy Analysis icon to display the Choose Domain dialog box. Select a domain, then click Finish. An object representing the domain will appear in the treeview. Right-click the object, and choose Perform Analysis. The program will ask you to choose a user and a machine. Choosing the machine determines the site, and therefore the site policies, as well as the machine and user policies. Choose a machine, and click OK.

The system will show a new object labeled *user at machine*—for example, Joe at Mypc. Under the new object, three objects will appear: User Hierarchy, Machine Hierarchy, and Resultant Policy. Open Resultant Policy, and you'll see a Settings object. Click Settings, and a Launch Group Policy Snap-In object will appear in the right-hand pane.

The snap-in object launches the standard Group Policy snap-in. Inside the Computer Configuration and User Configuration objects, you'll see the typical Group Policy subcategories, such as Software Settings and Windows Settings. Unfortunately, FAZAM 2000 RFV doesn't tell you that site policy X was in effect but policy Y overrode policy X. Instead, FAZAM 2000 RFV boils down the RSoP into one imaginary Group Policy Object (GPO), then uses the Group Policy snap-in to display that policy. You need to dig into the policy to determine which folders (e.g., Software Set-

tings, Windows Settings, Administrative Templates) actually have something in them. But at least FAZAM 2000 RFV gives you a start.

## Logoff.exe and Logoff.vbs

On average, I receive a letter a month asking whether a command-line tool is available to force a logoff. For a long time, I knew of no such tool. But the *Microsoft Windows NT Server 4.0 Resource Kit Supplement 4* contains two: a compiled program, logoff.exe; and a VBScript routine, logoff.vbs, that does even more than logoff.exe does.

You can call logoff.exe from a batch file. Its syntax is

```
Logoff [/n] [/f]
```

When you invoke logoff without options, logoff asks whether you really want to log off and accepts a y (yes) or n (no) response. When you type y in response, NT logs you off unless an application has unsaved data, in which case the application stops the logoff process and asks you to verify that you really want to log off without saving your data. Clearly, a logoff command that requires confirmation isn't much help in a batch file. Enter the /n and /f options.

If you specify the /n option, logoff doesn't present its confirmation prompt. However, if you have open and unsaved data, your applications still ask whether you really want to log off. You might want to retain those application prompts, because batch scripts that kick users off their machines and lose their data make for lousy administrator-user PR. But if you don't need the application prompts, you can use the /f option to force the logoff.

Logoff.exe is convenient because it's a standalone executable binary. But if you want to log off a user at a remote machine, you'll need to use the VBScript routine. You'll find both logoff.vbs and logoff.exe in the resource kit directory. So, when you type logoff from a command prompt in the resource kit directory, you'll execute logoff.exe, not logoff.vbs. (The command shell always looks for .exe files before it looks for script files.) To force the shell to run the VBScript routine, type the entire filename (i.e., logoff.vbs) at the command line.

Because Microsoft based logoff.vbs on Web-Based Enterprise Management (WBEM), you can use this script to force logoffs on remote machines. But the target machine on which you want to force the logoff must have WBEM installed. All Windows 2000 systems include WBEM; for NT machines, you can download it from Microsoft's Web site for free or get the WBEM base code from the resource kit.

The logoff.vbs syntax is

```
Logoff.vbs /s <servername> /u <username> /w <password> /f
```

You use the /s option to execute the script on a particular server. You can pass an administrative account name and password to that server with the /u and /w options, respectively. The /f option forces all applications to close. Logoff.vbs doesn't have anything like logoff.exe's /n option because logoff.vbs never prompts with an "Are you sure?" message.

Suppose I want to force the user sitting at a server named \\uptown to log off. Suppose also that I have an administrative account named BigGuy on \\uptown, and BigGuy's password is stingray. I could force a logoff with the command

```
Logoff.vbs /s \\uptown /u bigguy /w stingray /f
```

Embedding an administrative account name and password in a batch file isn't often a good idea, so you might be a bit limited in what you can do remotely with `logoff.vbs`. But if you ever do need to do a logoff from a batch file, you'll appreciate that `logoff.vbs` is readable code. You can read the VBScript file, then use the same technique to include logoff functionality in your own batch script. You've just got to love seeing Microsoft embrace open source.

## Netdom for Windows NT

Anyone who manages a large network knows that although Windows NT provides a broad suite of administrative tools, the tools' GUIs can be a pain. The fact that User Manager for Domains is a GUI tool is wonderful for first-time administrators, because they can leverage skills they learned playing Solitaire when they maintain their network. But User Manager for Domains isn't fit for administering hundreds of user accounts because you can't automate the tool's functions.

One administrative function that has always been difficult to automate is fixing broken trusts. However, the Netdom utility in the *Microsoft Windows NT Server 4.0 Resource Kit Supplement Two* can maintain trust relationships. Netdom lets you build new trust relationships and reset existing trusts from the command line.

Think about how you build trust relationships without Netdom. Suppose your network contains two domains—TRUSTED and TRUSTING—and you want to create a trust relationship that makes TRUSTING trust TRUSTED. To create this trust, you need an administrative account in the TRUSTING and TRUSTED domains. Log on to a TRUSTING domain controller (DC) with your TRUSTING administrative account, and log on to a TRUSTED DC with your TRUSTED administrative account. Then, fire up User Manager for Domains, point the tool at the TRUSTED domain, and tell User Manager for Domains that TRUSTING can trust TRUSTED. Refocus User Manager for Domains on the TRUSTING domain, and NT sets TRUSTING to trust TRUSTED. Whew!

Netdom's approach is easier. Like User Manager for Domains, Netdom requires you to have two administrative accounts, one in TRUSTED and one in TRUSTING. Netdom sometimes becomes confused if your username in TRUSTED is the same as your username in TRUSTING and the two accounts have different passwords. I recommend using different account names in the two domains or using accounts with identical names and identical passwords.

Netdom accepts the username and password for your TRUSTING account but not for your TRUSTED account—I'm not sure why Netdom has this discrepancy. However, you can use the old Net Use ... IPC\$ trick to establish your credentials in the TRUSTED domain. Just type

```
net use \\<name_of_PDC_in_TRUSTED_domain>\IPC$ /user:TRUSTED\<your_username>
```

Or you can run Netdom from a domain administrator account in TRUSTED, in which case you don't need to use Net Use to connect to the IPC\$ share.

Suppose the name of your administrative account in TRUSTING is `admin` and the account's password is `swordfish`. If you're logged on as a TRUSTED administrator, you make TRUSTING trust TRUSTED by typing

```
netdom /domain:TRUSTING /user:TRUSTING\admin /password:swordfish master TRUSTED /trust
```

That's a long command line; it boils down to

```
netdom <info_about_the_trusting_domain> master <name_of_the_trusted_domain> /trust
```

You might be thinking, “So what? I rarely build trusts.” Remember that you can run Netdom to do more than just build trust relationships; you can use the utility to *rebuild* trust relationships. If you come to work one morning and find DCs complaining that they can’t establish a link with a trusted domain, what do you do? Until now, your best option was to reboot the DC—not a great answer for a production server. Your worst option was to rebuild the trust relationship. Now, Netdom offers a better solution than either of those: Run Netdom /trust to rebuild an existing trust relationship in a flash. As a bonus, Netdom breaks trust relationships, too. For example, type

```
netdom /domain:TRUSTING /user:TRUSTING\admin /password:swordfish master
TRUSTED /delete
```

### **Netdom’s Member Option**

The *Microsoft Windows NT Server 4.0 Resource Kit Supplement Three* includes improvements to some old friends, including Netdom. The member option of Netdom’s Supplement Three version makes managing many machine accounts easy.

### **Adding Accounts to a Domain**

One common method for getting a machine to join a domain involves using Server Manager to create a machine account, then sitting down at the machine you want to add and telling it to join the domain. Another method requires you to start the target computer, tell the system’s Setup program to join the domain, and enter a domain administrator’s username and password. Neither of these processes lets you work remotely, so neither is ideal for adding machine accounts in bulk. To tell machine B to join domain C while you’re sitting at machine A, you need Netdom’s member option.

Suppose you want a computer named Fido to be a member of a domain named Browsers. Log on to any NT machine that has the resource kit. Open a command prompt, and type

```
netdom /domain:browsers member \\fido /joindomain
```

This command line tells your NT machine to find the PDC for Browsers and create a machine account for Fido, then locate Fido and instruct Fido to log on to Browsers the next time Fido powers up. If the machine you’re working at is a member of the Browsers domain, you can leave off the /domain: parameter.

Netdom has one requirement: You must be a domain administrator for Browsers and a local administrator for Fido. Suppose Fido is a functional NT workstation that hasn’t joined a domain, and you’re trying to issue the Netdom command for Fido from another machine (I’ll call it Mastiff). The first part of the Netdom command locates the PDC and asks it to create a machine account for Fido. If you logged on to Mastiff with a domain administrator account, the PDC creates a machine account for Fido without a problem. Then, the second part of the Netdom command contacts Fido and instructs Fido to join Browsers. But before Fido follows your command to join the domain, it must recognize you as a local administrator. You need to establish your administrative credentials with Fido before you use Netdom. If Fido has a local administrative account named Marco and that account has the password polo, you can establish those credentials by typing

```
net use \\fido\ipc$ /user:marco polo
```

before you type the Netdom command line.

### Other Netdom Member Commands

Netdom's member option lets you do some other neat things. Suppose you want to detach Fido from Browsers. You need to tell Fido to stop logging on to Browsers and delete Fido's machine account. The command line

```
netdom member \\fido /joinworkgroup browsers
```

tells Fido that it's no longer a member of a domain; instead, it's a member of a workgroup named Browsers. (You don't need to give the workgroup the same name as the domain.) The command line

```
netdom member \\fido /delete
```

tells NT to find a PDC and zap Fido's account.

In addition, you can use the command line

```
netdom member \\fido /query
```

to find out Fido's domain-membership status. And

```
netdom member \\fido /add
```

tells NT to create a machine account for Fido in Browsers but doesn't tell Fido to log on to the domain under that account. Netdom lets you do things from the command line that used to require a lot of walking around.

## Netdom 2000

Because Windows 2000 supports larger domains than Windows NT 4.0 supports, many administrators will want to reduce the number of domains in their enterprise. Unfortunately, Win2K doesn't include a domain-consolidation tool to help you achieve this goal. However, you can use the *Microsoft Windows 2000 Resource Kit's* Netdom tool to consolidate an NT 4.0 resource domain into a Win2K domain.

Suppose you decide to simplify your domain structure by replacing a resource domain with an organizational unit (OU) in what was once a master domain, moving NT accounts from the resource domain to the OU. A separate OU accomplishes many of the same tasks a resource domain accomplishes because you can cede control of the OU to a departmental group of administrators who then have local control.

Creating the OU, establishing the group of administrators, and giving the group power over the OU are simple tasks that you can accomplish quickly. But to complete the process, you must move all the workstations' and servers' machine accounts to the OU. In addition, you need to change the machines' domain affiliations if the machines belong to another domain. This step requires you to visit each machine—a job no administrator wants.

Netdom lets you move a machine's domain and OU affiliation without visiting the machine. The command line looks like

```
netdom move /d: /uo: /po: /ud: /pd: /reboot: /ou: /verbose
```

To put a machine in a domain, you need an administrator account that the machine recognizes (so that the machine lets you put it into a domain) and an administrator account that the domain recognizes (so that the domain lets you create the machine account). The `/uo:` and `/po:` options identify the account name and password for an administrator account on the local machine. However, if the machine already belongs to a domain, a domain administrator account from that domain is sufficient. The `/ud:` and `/pd:` options identify the account name and password for an administrator account on the domain that you're joining. For either account, you can use the `domain\name` construction (e.g., `bigdogs\Joan`). If the account exists on a domain (as opposed to being a local SAM account), you can use the more recent email-like construction (e.g., `joan@bigdogs.com`). The `/reboot:` option remotely reboots the machine that you're moving to the domain and OU.

You need to refer to a machine by its NetBIOS name rather than by its DNS-like name. For example, if you wanted to move the machine `lemon.fruit.com` to the domain `citrus.com`, the command would be

```
netdom move /d:citrus.com lemon
```

rather than

```
netdom move /d:citrus.com lemon.fruit.com
```

Suppose you want to move the machine `lemon` to the domain `fruit.com`. `Lemon` has a local administrator account with a blank password, and `fruit.com` has an administrator account named `carmen`, with the password `ascorbic`. The domain `fruit.com` contains an OU called `citrus` that you want to put the `lemon` account into. Finally, you want `lemon` to reboot 5 seconds after the command finishes so that the changes will take effect. The `Netdom` command for this action is

```
netdom move /d:fruit.com lemon /uo:lemon\administrator /po:""
      /ud:carmen@fruit.com /pd:ascorbic /ou:citrus /reboot:5 /verbose
```

The `/verbose` option helps you determine what went wrong if the command fails.

Although you must enter the entire `Netdom` command for each machine, you could probably use `VBScript` to automate this task. More troubling than this minor inconvenience is that `Netdom` doesn't work with `Win2K beta 3`—the `/uo:` option and remote capabilities aren't functional. However, Microsoft promises to fix these problems by the time `Win2K` ships.

### ***Netdom 2000's Trust***

`Netdom` lets you create, destroy, and monitor trust relationships; add machines to a domain; remove machines from a domain; and synchronize system times. That kind of command-line power is great, except for one caveat: `Netdom` is a nightmare of command-line options and idiosyncrasies. Here, then, is `Netdom` simplified.

`Netdom`'s syntax is

```
netdom <command>
```

where the possible commands are `Add`, `Join`, `Move`, `Remove`, `Reset`, `Verify`, `Trust`, `Query`, and `Time`. For information about these commands' functions, see the resource kit's `Netdom` Help files. In this section, I focus on the `Trust` command.

In general, when you join two components (i.e., create a trust relationship between two domains or join a machine to a domain), one of those components needs to recognize you as an administrator. Netdom arbitrarily calls one component the *object* and the other the *domain*. As I explained, you use the /uo: and /po: options to inform Netdom of the account name and password of an account that the object recognizes as an administrative account. You do the same for the domain with the /ud: and /pd: options. When you join (or disconnect) a machine to a domain, the machine is the object and the domain is the domain. However, when you set up a trust relationship in which a trusting domain will trust a trusted domain, then the trusting domain becomes the object and the trusted domain becomes the domain. Thus, you'd use the /uo: and /po: options to specify the administrative account for the trusting domain, and you'd use the /ud: and /pd: options to specify the trusted domain's administrator.

In the same way, you name the domain with the /d: option and follow the /d: parameter with the object's name. Therefore,

```
netdom trust /d:B.com A.com ...
```

starts off the command to make domain A.com trust domain B.com.

You create a trust relationship by adding the /add option. Suppose you want domain A.com to trust domain B.com. The Netdom command would look like

```
netdom trust /d:B.com A.com /add /uo:admina@a.com /po:pwa /ud:adminb@b.com
/pd:pwb /verbose
```

where A.com's administrator is admina and B.com's is adminb, with passwords pwa and pwb, respectively. I always use the /verbose option. Building a Netdom command can be complex, and /verbose helps me figure out where I made a mistake if Netdom doesn't work.

You can build two-way trusts by adding the optional /twoway command. In this case, the object and domain classifications don't matter:

```
netdom trust /d:B.com A.com /add /uo:admina@a.com /po:pwa /ud:adminb@b.com
/pd:pwb /verbose /twoway
```

You break a trust relationship by replacing /add with /remove. If you've created a two-way trust, you need to include the /twoway option when you break the trust, as this example shows:

```
netdom trust /d:B.com A.com /remove /uo:admina@a.com /po:pwa /ud:adminb@b.com
/pd:pwb /verbose /twoway
```

Netdom's /verify and /reset commands are useful in monitoring trust relationships. Win2K will help reduce the number of trust relationships in an enterprise, but as long as trusts are necessary, Netdom is invaluable.

## Netest

Trust can be so hard to keep. Relationships break down. If you have many trusts in your Windows NT enterprise network, you're likely to find that some pair of domains that should trust one another don't. NetLogon, the service that provides secure NT-to-NT communications, has failed. NetLogon communications serve three important relationships: connections between an NT

machine and its domain controller (DC—adding a machine to a domain establishes a kind of trust relationship), connections between PDCs and BDCs (synchronizing the domain’s SAM database requires a trust link), and standard, domain-to-domain trust relationships. Any one of these links can dissolve, causing mysterious problems.

The *Microsoft Windows NT Server Resource Kit* and *Microsoft Windows NT Workstation Resource Kit* include a tool, `nltest.exe`, that lets you quickly test the status of the NetLogon linkages between machines. This utility usually can’t repair trust relationships, but because `Nltest` is a command-line tool, you can incorporate it into batch files to automatically monitor machine connections.

To test a machine’s link to its domain, you use `Nltest`’s `/query` option, which verifies that NetLogon is running. For example, if you have an NT server (or NT workstation) named `MINBAR` and want to see if NetLogon is running and functioning properly on that machine, open a command line and type

```
nltest /server:minbar /query
```

You’ll see a few messages and finally the *Command completed successfully* message.

Each active domain member should have a functioning secure channel to a DC. To check that status, you use the `/sc_query` option. If `MINBAR` is a member of a domain named `B5`, you test its domain connection with

```
nltest /server:minbar /sc_query:B5
```

The command responds with success or failure, and provides the name of the DC that `MINBAR` has a secure channel to. If the command reports a problem, you can replace `/sc_query` with `/sc_reset` to try to reset the secure connection. The `/sc_reset` option might also work to reset a broken trust relationship.

NetLogon also governs PDC-BDC communications. You can find out what machines are DCs on a domain with the `/dclist` option. For the `B5` domain example, you can list the DCs with

```
nltest /dclist:b5
```

You can get the name of the PDC with `/dcname`.

You can control PDC-BDC SAM replications with the `/repl` option or resynchronize the entire SAM database with the `/sync` option. For example, if you have a BDC named `AJAX`, you can force that BDC to dump its copy of the domain’s SAM and request a new one from the domain’s PDC with

```
nltest /server:ajax /sync
```

To tell the BDC to request the changes to the SAM since the last replication, replace `/sync` with `/repl`. If that domain has a PDC named `XERXES`, you can initiate the process from the PDC’s side with

```
nltest /server:xerxes /pdc_repl
```

Domain-domain trust relationships get a little tricky. The `/trusted_domains` option shows you what domains are trusted by the domain that your machine is in. Suppose you have a two-domain

enterprise with domains MASTER and RESOURCE. All the user accounts are in MASTER, and all the NT machines are members of domain RESOURCE. You've established a trust relationship so that RESOURCE trusts MASTER. You're logged on to a machine with your user account, which lives in domain MASTER, and the machine you're logged on to is a member of domain RESOURCE. If you run

```
nltest /trusted_domains
```

you'll be told that MASTER is trusted. The fact that you're logged on as a member of MASTER is irrelevant. The message means that RESOURCE, the machine's domain, trusts MASTER. If you logged on to the DC at MASTER and ran the same command, you'd get a blank list.

Nltest lets you determine whether you can establish a NetLogon session with a particular machine. If NetLogon is up, you can use the /sc\_query option to test connections to a DC and the /sc\_reset option to try to repair a link. The remaining options let you examine a machine's link to its domain, PDC-BDC connections, and trust relationships.

## Rcmd

Have you ever wanted to open a command prompt window on one computer and work at the command prompt of a computer elsewhere on the network? The *Microsoft Windows NT Server 4.0 Resource Kit* has some great tools that let you do from the command line what NT lets you do from only the GUI. For example, Addusers lets you modify user accounts and group membership; Netdom lets you change a workstation's domain membership and create and destroy trust relationships; Dhcpcmd lets you control your DHCP server and scopes; Winscl administers a WINS server; Rmtshare creates, destroys, and modifies permissions on file shares; and Xcacls does the same for NTFS permissions.

However, most of these tools can control only the machine you run them on. If you want to use Netdom to move a workstation's membership from one domain to another, you must walk over to that workstation, sit down, and run Netdom locally. In addition, Netdom is the primary in-the-box tool for moving NT 4.0 resource domains into Windows 2000 organizational units (OUs). Therefore, running some of these tools remotely is a real benefit.

The resource kit's Telnet service is useful, but it isn't secure. Someone sniffing network packets can intercept your password as you log on to the Telnet server. Here, I describe an alternative service that uses NT authentication: Remote Command (Rcmd). This service has two parts: the server portion and the client portion. Microsoft includes the Rcmd service with the server resource kit but not with the workstation resource kit. (I strongly recommend that you obtain Supplement Three of the resource kit.)

Microsoft built the server portion of Rcmd as a standard NT service that consists of just two files: rcmdsvc.exe and oemnsvrc.inf. You can find these files in the directory you installed the resource kit in. To install the service, copy these two files to the \winnt\system32 directory of the server you want to control. Then, open the Control Panel Network applet. Click the Services tab to show the network services (e.g., Server, DHCP), then click Add. After NT prepares its list of possible network services, you'll see the new option, Remote Command Server. Choose that option, then click OK. Control Panel will want to rebind the network services, so you'll need to reboot the system. Before you do, configure

the Remote Command Server to start automatically. Close the Network applet, and open the Services applet. Scroll down the list, and you'll see that you now have a service called Remote Command Server. Click this entry, then click Startup. Choose Automatic to have the service start automatically when you boot.

After you reboot, look in the directory you put the resource kit files in for the program `rcmd.exe`. Copy this program to your workstation. (The resource kit license lets you copy resource kit files to all computers in one location.) You can use `rcmd.exe` in several ways. To run one command on a remote server, type

```
rcmd \\<servername> <command>
```

For example, the command

```
rcmd \\ignatz dir
```

opens a window and displays the entire `\system32` directory from a remote server named Ignatz. As soon as the command is finished, the window closes. Typing

```
rcmd
```

causes the Rcmd service to prompt you to enter the name of a server; type the name without the backslashes (e.g., Ignatz rather than `\\Ignatz`). A command window opens, and you can type in as many commands as you like. The command

```
exit
```

closes the window. After you exit, Rcmd prompts you for another server name. Press Ctrl+C to shut off Rcmd.

Microsoft maintains that you can perform every administrative task in Win2K from the command line. If you learn Rcmd, you'll be better able to use Win2K.

## Repadmin

### *Get a Handle on AD Internals*

With luck, you'll never have to worry about Active Directory's (AD's) innards—every domain controller (DC) will replicate smoothly to every other DC. But should AD ever act oddly, you'll need tools to help diagnose the problem. The *Microsoft Windows 2000 Server Resource Kit* includes one such tool: `repadmin.exe`.

Repadmin can determine which machines a given DC replicates with. In domains of six or fewer DCs, the DCs form a bidirectional ring for replication, so every DC in the ring has a partner to its "left" and its "right." (More than six DCs leads to a more complex mesh structure.) To identify those partner DCs, type

```
repadmin /showreps <DCname>
```

where *DCname* is the DNS name of the DC whose partners you want to know. You'll get some fairly lengthy output—Figure 1 shows an excerpt.

**Figure 1:***An excerpt of Repadmin output*

```

DSA Options : IS_GC
objectGuid : c735 fab4-d564-4a2a-9dc0-225a232d71cf
.
.
.
INBOUND NEIGHBORS
.
.
.
DC=uptown,DC=acme,DC=com
  Headquarters\DS2 via RPC
    objectGuid: cf78b375-063a-4c5b-899f-8caff22c7f36
    Last attempt @ 2001-03-17 19:57.52 was successful.
  Headquarters\DS3 via RPC
    objectGuid: 8e90169a-dbf4-461b-97f5-1535085b9c04
    Last attempt @ 2001-03-17 20:04.56 was successful.
CN=Configuration,DC=acme,DC=com

```

Figure 1's DSA Options line tells you that this server is a Global Catalog (GC) server. The objectGuid line identifies the DC by its globally unique identifier (GUID) rather than its DNS name. (If you ever need to find a DNS name from a DC's GUID, open the Forward Lookup Zones folder in the Microsoft Management Console—MMC—DNS snap-in. In the folder for your AD domain, open the \_msdcs folder. For each DC in the domain, you'll see a record that contains the DC's GUID and DNS name.) Repadmin also documents the most recent replication attempts. Figure 1 shows that the DC successfully replicated to both its partners the last time that it tried.

The complete output also includes basic forest information. The configuration naming context lists the domains, sites, and DCs in the forest. Naming context is Lightweight Directory Access Protocol (LDAP) terminology for "information that AD needs to replicate." Repadmin also provides the schema naming context—a list of the fields in the AD database. Finally, Repadmin lists the domain naming context—the actual user and machine accounts and any other information the AD stores.

By default, Repadmin reports on your DC's replication partners for all three naming contexts. To limit output to a specific domain's domain naming context, include the LDAP terminology for that naming context in the command

```
repadmin /showreps <naming-context> <DCname>
```

To build the naming context, prefix each piece of the DNS name with dc=. So, to find out about the ds1.uptown.acme.com DC's replication partners, you'd type

```
repadmin /showreps dc=uptown,dc=acme,dc=com ds1.uptown.acme.com
```

## ***Forcing AD Replication***

In the previous section, I explained how to use the Win2K Server resource kit's Repadmin utility to find a DC's replication partners and how to use the required LDAP terminology to phrase your queries. Repadmin also lets you force those partners to synchronize.

Suppose I have two DCs—ds1 and ds2—both of which are members of acme.com. I want to run a script that works best when DCs have consistent copies of AD, so I want to synchronize ds1's and ds2's copies of the directory. Because AD replication is pull-only, synchronizing two DCs requires two events: I need to direct ds1 to pull ds2's changes, then I need to tell ds2 to pull ds1's changes. Both events are necessary because other DCs might have modified ds1's or ds2's copy of AD.

Use the following syntax to tell one DC to request updates from another DC:

```
repadmin /sync <namingcontext> <destinationDCname> <sourceDCGUID> /force
```

*Namingcontext* is LDAP-ese for the particular database that you want to replicate. For example, LDAP's naming context for the uptown.acme.com AD domain is dc=uptown,dc=acme,dc=com. *DestinationDCname* is the DNS name of the pulling DC. To tell ds1 to pull AD changes from ds2, I replace *destinationDCname* with ds1.acme.com.

The odd parameter is *sourceDCGUID*. In its place, you need to put the GUID of the DC that ds1 should pull changes from. You can use the Repadmin /showreps command to obtain that information. One of the first lines in that command's output resembles the line of output in Figure 2. The string that follows the colon is the specified DC's GUID. If that DC were ds2, we could use the command that Listing 1 shows to tell ds1 to pull replication information about the acme.com domain from ds2. The response (e.g., *Sync from 8e90169a-dbf4-461b-97f5-1535085b9c04 to dun.win2ktest.com completed successfully*) tells you whether replication was successful.

### Figure 2:

*A sample objectGuid line from Repadmin output*

```
objectGuid : 8e90169a-dbf4-461b-97f5-1535085b9c04
```

### Listing 1:

*Replicating with a Specific DC*

```
repadmin /sync dc=acme,dc=com ds1.acme.com 8e90169a-dbf4-461b-97f5-1535085b9c04 /force
```

An easier approach uses an option that doesn't require any GUIDs. The /syncall option forces the DC to synchronize with all its partners. The syntax is

```
repadmin /syncall <destinationDCname> <namingcontext> /force
```

To force ds1 to pull changes from all its replication partners, you'd type

```
repadmin /syncall ds1.acme.com dc=acme,dc=com /force
```

If you omit the naming context, the DC replicates the forest's schema and configuration naming contexts.

Within 15 minutes after you bring a new DC online, it will run the internal Knowledge Consistency Checker (KCC) routine to choose replication partners. If you don't want to wait, you can use the following command to force the DC to run the KCC:

```
repadmin /kcc <DCname>
```

where *DCname* is the DC's DNS name.

## Tracking AD Replication

Administrators in a Windows NT 4.0 or mixed-mode AD domain must connect to the PDC before modifying a user account or other domain data. But administrators in a Win2K native-mode (i.e., AD) environment can update AD from any DC. Thus, native mode is more flexible than NT or mixed mode, but native-mode AD replication is more complex and implements some new concepts, two of which are update sequence numbers (USNs) and a high-watermark table.

Every DC keeps track of how many AD updates have occurred in its lifetime and gives each update a unique, sequential USN. Let's say we create a new, empty AD on a DC named dc1.acme.com. Then, we create an account named Mary on the DC. That account is the first AD update, so AD gives that update a USN of 1. (I'm greatly simplifying the USN numbering scheme, of course.) If we next create an account named John, AD assigns USN 2 to that update. Then, Mary updates AD by changing her password, which becomes USN 3, and John changes his password, making USN 4—the highest USN on this copy of AD.

Now, let's bring online a second DC, dc2.acme.com. As part of the process of becoming a DC, DC2 copies DC1's AD database. But DC2 doesn't copy AD's entire history; it copies just the latest information. Thus, DC2 copies the two user accounts and considers each one an update, so DC2's USN is 2.

DCs use their own and other DCs' USNs to determine whether DCs are up-to-date with one another. For example, when DC2 asks DC1 for updates, DC2 doesn't want a tiresome litany of everything that DC1 has ever learned; instead, DC2 simply wants to know what has changed since the last replication. In effect, DC2 says to DC1, "Tell me everything that's happened since your USN 4." As you can see, DC2 must remember the last USN it got from DC1; that USN is called DC1's high watermark. Each DC keeps a table of the high watermarks from all its replication partners. You can use the Repadmin /showreps command to view that table:

```
Repadmin /showreps <namingcontext> <dcname> /verbose
```

For example, to see DC2's high-watermark table, you'd type

```
Repadmin /showreps dc=acme,dc=com dc2.acme.com /verbose
```

Excerpted, the output might resemble

```
=== INBOUND NEIGHBORS ===...
Site1\DC1 via RPC...
  USNs: 59228/OU, 59228/PU
```

You'll see a line for each replication partner. The high-watermark value is the number that follows USNs:—59228, in the example.

How can you use this information? In a properly functioning AD, you wouldn't ever have to. But suppose you've given a user some new permission or right, and the user is sometimes unable to exercise that permission or right. What might the problem be? Most likely, one of the DCs in your network hasn't received the updates detailing the new right or permission.

To determine whether that's the problem, open a command line at the user's desk and use the Set command to determine which DC the user is logged on to. Then, retrieve the high-watermark table for that DC. You'll be able to see whether the DC that logged the user on is behind the times compared with the other DCs. If it is, force a replication following the instructions under "Forcing AD Replication," and the problem should disappear.

### ***AD Replication and Up-to-Date Vectors***

In the previous section, I explained how AD uses USNs to track changes to each DC's copy of AD's contents. Each server remembers the most recent USN that the server obtained from its replication partners (i.e., the high watermark) and uses that USN to control replication. For example, DC1 might say to DC2, "Tell me what updates you've made since your USN 10,000."

When an administrator creates a new user account on DC1, that update increments DC1's USN, causing DC2 to request the AD changes associated with that update. But when DC2 incorporates that update and increments its own USN, DC2's new USN leads DC1 to request an update from DC2—including the new user account, which DC1 already knows about. Recording that update would lead to an endless cycle of rerecording the same piece of information.

To prevent changes from cycling indefinitely, AD remembers not only the local USN that generates an update but also the GUID of the DC that originated the update. Thus, AD can differentiate between an update that an originating DC generates and one that a DC merely passes to another DC. Every DC remembers not only the latest USN received from each replication partner but also the USN associated with the most recent update originating from every other DC in the domain. The table that contains the latest originating USNs is called the *up-to-date vector*.

DCs use the up-to-date vector to prevent infinite update loops. Suppose that the last originating update that DC2 received from DC1 was USN 9871 and the last update that DC2 originated was USN 1500. DC2 says to DC1, "Tell me about the updates you've received since you last updated me, but skip any that originated with DC1 and had a USN of 9871 or less and any that originated with DC2 and had a USN of 1500 or less." Suppose that DC1 had one new update, with a USN of 10,001. When DC1 sends that update to DC2, DC2 saves the information as USN 1553 but also remembers that the originating USN was 10,001 and that the update originated on DC1. DC2 increases its up-to-date vector entry for DC1 to 10,001. (DC1 also modifies its up-to-date vector to show that DC1's up-to-date vector is 10,001.)

When DC1 next asks DC2 for updates, DC1 says "I know about your updates through your USN 1552, and I've seen every update that originated with DC1 through USN 10,001 and every update that originated with DC2 through USN 1500. Do you have any newer updates?" DC2 sees that it has a USN greater than 1552, but that the originating DC was DC1 and the originating DC's USN was 10,001. Because DC1's up-to-date vector includes all DC1's originating entries through USN 10,001, DC1 knows about DC2's USN 1553 and doesn't record a new update.

You can use the Repadmin /showvector command to view a DC's up-to-date vector. To dump the vector for dc1.acme.com in a domain called acme.com, you'd type

```
repadmin /showvector dc=acme,dc=com dc1.acme.com
```

Figure 3 shows some sample output. Note that the last line shows a GUID rather than a DC name: After you decommission a DC, its entries still appear in other DCs' up-to-date vectors, but the entries appear under the former DC's GUID rather than a DC name.

**Figure 3:***Sample output from the Repadmin /showvector command*

```
Site1\DC1 @ USN 21595
Site1\DC2 @ USN 162530
Site2\DC3 @ USN 118741
c4c0767b-d99d-474c-bfcf-8e562d43d0e7 @ USN 73180
```

## Rmtshare

Suppose you're building a disaster-recovery script. You have everything about your Windows NT installation scripted perfectly, except for the creation of shares. Building shares and setting share permissions is simple from Windows NT Explorer, but you want to build shares from a batch file. The *Microsoft Windows NT Server 4.0 Resource Kit's* Rmtshare utility lets you create and delete file shares, modify the shares' permissions, and display information about the shares. Rmtshare lets you create and modify shares on remote machines, but the tool doesn't work for print shares.

To create a share with Rmtshare, type

```
rmtshare \\<server name>\<share name>=<drive:\directory>
```

The *server name* variable is the name of the machine that the share resides on. *Share name* is the name that you want to give the new share. And *drive:\directory* is the path to the directory you want to share. Suppose you want to share a directory called C:\finance that resides on a machine named Wally, and you want to call the share Bucksdata. Log on to any machine with an account that Wally recognizes as administrative, and type

```
rmtshare \\wally\bucksdata=c:finance
```

After you type this command line, you'll see the message *The command completed successfully*. At that point, the share is immediately available. Permissions on the share are Full Control for the Everyone group.

You can then use Rmtshare's /grant option to restrict users' permissions on Bucksdata. A /grant command line looks like

```
rmtshare \\<server name>\<share name> /grant
<username>:<permission>
```

This command line is simple, but it isn't quite as simple as it looks. The *username* variable obviously refers to a username. But if you're sitting at machine A and instructing machine B to change permissions for a user named Jennifer, machine B might not know whether you want to change Jennifer's account in machine A's SAM database, her account in machine B's SAM database, or her domain account. If you're referring to Jennifer's domain account and the domain's name is Corporate, I recommend using corporate\jennifer as the /grant command line's username.

The /grant command line's *permission* value can be fc, which stands for Full Control; r, which specifies Read Only rights; c, which gives the user Change access; or n, which specifies No Access. Suppose I have two users with accounts in the Corporate domain. The accounts' usernames are jennifer and joe. I want to change the accounts' permissions on the Bucksdata share on Wally; I want Jennifer to have Change access and Joe to have No Access. I need to type

```
rmtshare \\wally\bucksdata /grant
    corporate\jennifer:c
    corporate\joe:n
```

Rmtshare has a couple of other notable options. The /remove option lets you remove all references to a user from the share's permissions list. To remove Joe's No Access designation, type

```
rmtshare \\wally\bucksdata /remove corporate\joe
```

Rmtshare's /delete option lets you stop sharing a directory:

```
rmtshare \\wally\bucksdata /delete
```

And finally, you can use Rmtshare with just a server name and share name to find out information about a particular share. Type

```
rmtshare \\wally\bucksdata
```

to see detailed information about Bucksdata, including the share's path, its permissions, and any comment you've added to the share. Or use the command line without \bucksdata to list all the shares on Wally, including the hidden shares.

For command-line aficionados, Rmtshare fills an important NT gap. If only Rmtshare managed printer shares!

## Setprfdc

Windows NT 4.0 Service Pack 4 (SP4) contains Setprfdc, a cool utility that lets you adjust secure connections between Windows NT machines. Setprfdc isn't in any *Microsoft Windows NT Resource Kit*, but it's a utility you don't want to miss out on.

Because NT is a secure OS, NT machines must perform many authentications each day. On domain-based NT networks, computers must find a domain controller (DC) for those authentications. Which DC does an NT machine use? The machine logon process involves finding a DC; the computer uses that DC for subsequent authentications until a user reboots the machine. This link is a secure remote procedure call, or secure RPC.

Unfortunately, that machine logon might not be the beginning of a beautiful friendship. Sometimes busy local DCs force NT machines to find authentication buddies far away. For example, users at a large firm in Austin might find one day that their computers' DCs are in Paris. One long-distance authentication isn't a problem, but if an Austin workstation requires a lot of authentication traffic, that traffic could stress the bandwidth of a slow transatlantic WAN link. The firm's administrators would undoubtedly be happy to hear about Setprfdc, a utility that lets you specify a new authentication buddy for a running NT machine.

The utility's syntax looks like

```
setprfdc <domain_name> <first_domain_controller>,<second_domain_controller>,<third_domain_controller>
```

Austin users whose systems are in the TEXAS domain and whose local DCs are named TX1 and TX2 can open a command line and type

```
setprfdc TEXAS TX1,TX2
```

The workstations will attempt to connect to TX1 first; if that connection attempt fails, they'll attempt to connect to TX2. If both connection attempts fail, the machines will maintain their current secure RPC to the Paris DC. If Setprfdc changes the users' connection, it will report which local DC the machine connected to.

Is this utility useful? Absolutely. Links similar to the authentication buddy connections exist between member servers and DCs, PDCs and BDCs, and DCs in trusting domains. Under some circumstances—such as WINS failures—DCs in one domain lose track of their authentication buddies in other domains. When such a situation arises, users face network delays at best and an inability to log on to the domain at worst. Suppose you arrive at your network operations center one morning and see an Event Viewer entry that indicates your network has lost its trust link to another domain. Do you start rebooting servers until the problem disappears? No. You just pull out your list of DCs in the trusted domain and use Setprfdc to dial for secure RPCs.

You can avoid this problem by finding each DC an authentication buddy on the same network segment; if WINS gets loopy, the DC can find its buddy via broadcasts. Or you can use LMHOSTS files to solve this problem.

Consider how you can use Setprfdc. Your domain's member servers, file and print servers, Web servers, and Exchange Server systems constantly need authentication help from a DC. If they become disconnected from their authentication buddy, they float around the enterprise looking for a new favorite DC. To prevent such a server from connecting to a faraway machine, place a Setprfdc command in an AT command to force your servers to establish secure RPCs with particular DCs.

## ShareUI

Let's look at a partial solution to an old problem: an all-in-one manager for file shares. The operant words are *partial solution*, though. Perhaps this field offers an opportunity for an enterprising utility developer.

Windows NT's GUI-based approach to file shares is simple and intuitive. You open My Computer or NT Explorer, right-click the directory you want to share, and select Sharing. A dialog box appears, you make a few choices, and in seconds you've set up a shared volume.

### ***A Problem Finding Shares***

Although this process is convenient, you'll eventually stumble upon a problem you wouldn't expect to have when managing a computer's file shares. You look at a server in Network Neighborhood, and you notice a share that you don't remember. Where, you might wonder, did that Datac share come from? You vaguely recall creating it a year ago for some temporary purpose, but you can't recall its value. You decide to get rid of it.

At the file server, you open My Computer and poke around the C and D drives, but you can't find a shared folder named Datac—there's no white hand with the blue sleeve on any of the folders. After some digging, you find a shared directory named C:\Files\Junk\Datac. How do you shorten this search process and quickly find a share path, given only its share name? One way is to use ShareUI.

### ***Solution: ShareUI***

In the directory in which you installed the *Microsoft Windows NT Server 4.0 Resource Kit*, you'll see the file `shareui.inf`. Right-click the file, and select Install. In a couple of seconds, you've installed ShareUI.

To see ShareUI in action, open My Computer. You'll see a new folder named Shared Directories. Open it, and you'll notice an array of white-hand icons, one for each share in the system, including the administrative shares. Double-click any icon, and you'll get a dialog box displaying the share's name. This dialog box will show you that elusive share path you've been searching for, as well as a User Limit adjustment box and a key that lets you view and modify the permissions (sadly, no NTFS permissions) on the share.

You can also use ShareUI to control shares on other machines from your computer. Navigate to the remote computer in Network Neighborhood, open the computer, and you'll find another Shared Directories folder. Open it, and you'll see that it looks identical to the folder you just looked at on the local computer. (ShareUI doesn't need to be running on the remote computer.)

### ***A Fly in the Soup***

I found an odd bug while I was examining an existing share's properties. I had a share whose local directory path was `F:\Presentations` that I'd created with the share name *presents*. When I used ShareUI to examine the share's properties, both the field that displayed the path (`F:\Presentations`) and the share name (*presents*) permitted editing. I easily changed the share's name from *presents* to *powerpoints*. The share's name changed immediately, and I had no problem using the new share name to connect to this renamed share from a remote system.

I then tried changing the path field from `F:\Presentations` to `F:\Stuff`, although no directory named Stuff existed on the F drive. ShareUI complained that it couldn't find the specified *file* (I think the error-message author meant *path*). I cancelled the operation. Oddly enough, canceling the operation removed the current *powerpoints* share. Not a huge bug, but an unexpected one.

### ***In Short***

Although ShareUI is a useful tool, it's missing a few components. NTFS permission control would be a helpful addition, as would a directory-browsing ability you could use when you create shares on remote machines.

## **Showmbrs and Diskquotas.pl**

Windows 2000's disk quotas don't let you assign different disk quotas to different groups of users—you can't give 100MB quotas to everyone in the Engineers group and 20MB quotas to everyone in the Managers group. But two *Microsoft Windows 2000 Server Resource Kit* tools give you a rudimentary solution that will let you type one line to set a disk quota for everyone in a group.

Showmbrs.exe lists the members of a specified group. You type

```
showmbrs <groupname>
```

to list the members of a local group and

```
showmbrs \\<domainname>\<groupname>
```

to list a domain's groups or a remote machine's local groups.

Diskquotas.pl is a Perl script that lets you create, modify, delete, copy, list, import, and export quota information about disk volumes. Options let you create a quota entry of a given size for a given user on a given volume. The syntax is

```
diskquotas.pl -create <drive>: -logonname <username> -setlimit <size> -silent
```

where *drive* is a drive letter, *username* is a username with or without a domain prefix, and *size* is an integer to which you append KB, MB, GB, TB, PB, or EB (to indicate kilobyte, megabyte, gigabyte, terabyte, petabyte, or exabyte, respectively). The -silent option tells diskquotas.pl to skip any prompts and just do what you tell it to do. If Bob is a member of the Acme domain and I want to set his disk quota to 500MB on the local server's D drive, I would type

```
diskquotas.pl -create d: -logonname acme\bob -setlimit 500MB
```

If Bob already has a quota entry, diskquotas.pl would ask me whether it should replace Bob's old quota value with the new value.

Clearly, diskquotas.pl is useful, but it's only a user-by-user tool. To set a quota for everyone in a group, you need to use Win2K's and Windows NT's powerful For command to capture the names of the group members with Showmbrs, then you need to feed that list of members to diskquotas.pl:

```
for /f "usebackq skip=2" %i in (`showmbrs <groupname>`) do diskquotas.pl -create  
  <drive>: -logonname %i -setlimit <size> -silent
```

Be sure to type the command as one line and substitute appropriate values for *groupname*, *drive*, and *size*.

Simplified, here's how For works. The command

```
for /f "usebackq skip=2" %i in (`firstcommand`) do secondcommand %i
```

executes the command named Firstcommand and feeds its output to a command called Secondcommand. The /f portion of the command string tells the For command to parse (i.e., return the first token on) each line of output Firstcommand produces as input to Secondcommand. Notice that I embed the first command within backward single quotes. The usebackq option tells the For command to treat the string within backward quotes as an executable command. The skip=2 parameter tells For to ignore the first two lines of Firstcommand's output—the first two lines of Showmbr's output are header lines.

The first %i in the command string specifies the variable For uses to store Firstcommand's output. You can give the variable any name prefixed by a percent sign. The second %i tells Secondcommand to use Firstcommand's output values. To run this command from a batch file, you must substitute %%i for both occurrences of %i in the command string.

All in all, this process is an ugly way to assign disk quotas to a group. But I hope you'll find it a useful one.

## SU

If you have the *Microsoft Windows NT Server Resource Kit* or the *Microsoft Windows NT Workstation Resource Kit*, you know that it's full of great-sounding utilities. But figuring out some of those utilities can be a problem. One utility worth figuring out is `su.exe`. It lets you start up a program under the guise of a different user.

### ***When this Utility Is Useful***

Suppose you're a network administrator in domain EARTH. You have two accounts on your network: CKENT and KALEL. One afternoon, you're logged on to your CKENT account, your ordinary user account, working on a monthly report in Microsoft Word. A user calls and says he's forgotten his password and asks you to reset it for him. No problem. You start up User Manager for Domains. You find his account and double-click on it, only to be told "Access Denied: the user properties cannot be edited or viewed at this time."

Then you remember you're logged on to your user account. So you'll have to shut down Word, log off CKENT, and log back on as KALEL, an account with administrative powers. But, you have a better idea: `su.exe`.

SU lets you start up User Manager for Domains (or any other application) under the KALEL account, even if you're not currently logged on as KALEL. You can then change the user's password, exit User Manager, and return to your Word document.

### ***Step by Step***

In its simplest form, an SU invocation looks like

```
su <name of account you want to use> <name of program> <domain of account>
```

In our example, all you have to do is open up a command line and type

```
su kalel usrmgr earth
```

SU will prompt you for the password for the KALEL account. Once you've entered the password, User Manager for Domains starts.

Suppose I have two user accounts, Mark (ordinary user account) and MarkA (administrative account), in domain ANDROMEDA. I'm logged on to my NT workstation as Mark and want to change my system's time. But, ordinary users can't change system times. To use the TIME command, I need a command line. So, I type

```
su MarkA cmd andromeda
```

I'm prompted for MarkA's password, I supply it, and I get a command prompt window. Then I can use the TIME command to change the computer's time.

### ***What Can Go Wrong***

SU is a neat utility, but you have to modify a user's rights before SU will work. If you just fire up a command line on a system that's installed with all default rights, the previous examples won't work. The CKENT account (the account running SU) must have two advanced user rights that NT

users of all stripes don't have by default: *Act as part of the operating system* and *Replace a process level token*.

An administrator can easily give those rights to CKENT. So, before you try SU the first time, log on to your NT machine with the administrative account—KALEL in this example—and open up User Manager for Domains.

If you're running User Manager for Domains, you'll need to direct User Manager to modify the rights granted on the machine you're working at. By default, User Manager for Domains modifies the rights that domain controllers (DCs), not machines in general, grant. For example, suppose you have two DCs, D1 and D2, and three member servers, S1, S2, and S3, and you're at a workstation named W1. All are NT machines that are members of a domain. If you're sitting at W1 logged on as CKENT and want to run SU, then CKENT must have the *Act as part of the operating system* and *Replace a process level token* rights on W1. So, again, before CKENT can run SU, you'll have to log on to W1 with an administrative account to grant CKENT those rights.

If you're using User Manager for Domains, you click User/Select Domain and type in W1. (Yes, W1 is a machine, not a domain, but that's how you control W1's rights.) Now click OK. But if you're just sitting at W1 and running the simple User Manager, you don't have to click User/Select Domain. Then click Policies/User Rights, and check the box labeled *Show advanced user rights*. Select *Act as a part of the operating system* and add CKENT's name. Do the same for *Replace a process level token*. CKENT can now run SU.

## Telnet Server

For the poor souls who remotely administer Windows NT networks, I have a real treat: a Telnet server. Remote NT administration is a great irritant compared with UNIX administration. Many of the GUI-based NT administration tools let you remotely administer other servers, but using these tools isn't practical on any line slower than 128Kbps (the tools are noticeably sluggish even over 128Kbps).

Consider having the User Manager program on machine A control machine B's user accounts: Machine A does a tremendous amount of work to request a hunk of data from machine B; machine A then massages and ships the data back to machine B. This process keeps both machines busy. A better solution is for machine B, the remote system, to do all the computing, shipping of graphical output, and sending of keystrokes back and forth—the same process Windows NT Server 4.0, Terminal Server Edition uses. WTS, however, is graphical. If you don't want delays and you're using a 28.8Kbps line, you need the simple, text-only interface that a Telnet server provides.

## Installing Telnet's Services

When you install the *Windows NT Server 4.0 Resource Kit* into a directory, the resource kit's Setup program creates another directory named Telnet. The Telnet directory contains two services called the Remote Session Manager and the Telnet daemon. You install them as you would install any network service: From Control Panel, Network, Services, click Add. When the system prompts you to pick a service, click Have Disk. Then you need to locate and point to the directory in which you installed the resource kit; this directory now has \Telnet appended to it. For example, if you told the resource kit to install to D:\Reskit, point Control Panel to D:\Reskit\Telnet. First, install the

Remote Service Manager service, and then, without rebooting, install Telnet Service Beta (Inbound Telnet). Then, reboot.

### ***Put to the Test***

To test the Telnet server, go to any machine on your network and click Start, Run; then in the text box, type

```
telnet <machine name>
```

where *machine name* is the name of the server on which you installed the Telnet server. The system prompts you for a username, and you must use explicit domains. If Janet is your username, for example, and the domain name is Dorians, specify dorians\janet (i.e., don't just specify janet). The system then prompts you for a password; enter the password, and you see a regular command prompt such as D:\Winnt>. Type any command-line command, and notice how quickly you get a response! Network overhead doesn't exist because the remote server does all the hard work and provides you with only a small amount of ASCII output.

The resource kit obviously provides needed relief from graphical tools for remote NT administration. Here are a few suggestions to replace your missing GUI tools. To administer user accounts, use Adduser. To control machine accounts and trust relationships, use Netdom. To create or delete file shares, or to modify their permissions, use Rmtshare. To control WINS, DHCP, or DNS servers, use Winscl, Dhcpcmd, and Dnscmd. To see which processes are running on a system and to stop runaway processes, use Tlist and Kill.

### ***Proceed with Caution***

Unfortunately, you'll pay a price for this cool tool: security. Telnet clients pass their text in clear text, so someone could easily sniff your password when you log on. So even though the Telnet daemon is neat, it's not for everyone. Although you could put PPTP on the server with the Telnet daemon and require everyone to PPTP into it, PPTP adds enough overhead to a server that you may end up losing more response time than you gain.

## **Winscl**

I gave a conference presentation that covered how WINS and DNS work, and conference attendees asked me, "How can I delete a specific record from a WINS database?" and "How can I dump a complete list of WINS registrations to an ASCII file?" The graphical WINS manager that ships with Windows NT can't perform either of these jobs, but a utility called winscl.exe, which comes with the *Microsoft Windows NT Server 4.0 Resource Kit* and *Microsoft Windows NT Server 3.51 Resource Kit*, can. Winscl offers much of the same functionality as the graphical WINS manager, and it provides a few functions the GUI program doesn't offer.

Despite its power, Winscl can be frustrating. I'm not exaggerating when I say that it makes Edlin's UI look friendly. To start Winscl, you type

```
winscl
```

on a command line, and the utility greets you with

```
TCP/IP or named pipe. Enter 1 for TCP/IP -
```

Winscl is asking you to specify whether you want to enter an IP address or NetBIOS name to identify the WINS server you want to use Winscl to control. If you enter 1, Winscl will ask for the server's IP address (such as my WINS server's address, 10.10.10.21). If you enter 0, it'll ask for the machine's Universal Naming Convention (UNC) name, which is your cue to enter the server's name preceded by two backslashes (e.g., \\WINS01).

After you select a server, Winscl produces a description of each of its 31 commands. By default, it lists all 31 commands after it completes every job. If you enter the NOME (no menu) command, Winscl won't display its menu of commands again until you exit and restart the utility. Another Winscl quirk is that you must capitalize all commands.

### **Winscl Commands**

Winscl's two most useful commands are DN (delete name) and GRBV (get records by version numbers). DN helps you clean out a WINS database. GRBV lets you dump all of a WINS server's records to an ASCII file.

Several months ago, someone registered with my WINS server as JOE<03>. I used Winscl to remove Joe from the WINS database. My dialog with my WINS server follows.

```
Command -- DN
Name? JOE
Do you want to input a 16 char (1 for yes) -- 1
16th char in hex -- 03
Scope 1 for yes -- 0
Status returned is (SUCCESS -- 0)
```

The other useful Winscl command, GRBV, lets you extract WINS records for storage in an ASCII file based on the records' *version numbers*, the Microsoft term for the order in which they were created. (The first WINS record that a WINS database creates is version number X, the next record's version number is X+1, and so on.) GRBV is convenient, but it doesn't give up the data easily. You get a bit of interrogation along the way, as the following dialog demonstrates.

```
Command -- GRBV
Address of owner WINS? 10.10.10.21
Want to specify range -- (input 1) or all (default) --
Use filter (1 for yes) --
Put records in wins.rec file (1 for yes) -- 1
Status returned is (SUCCESS -- 0)
```

If this example looks like too much work, you can create an ASCII text file to automatically answer Winscl's questions. Enter one answer per line for the Winscl dialog. For example:

```
1
10.10.10.21
GRBV
10.10.10.21
0
0
1
EX
```

EX is the command to exit Winscl. Name your ASCII file WINSIN.TXT, and invoke it by typing

```
winscl <WINSIN.TXT
```

The text dump process will run automatically.

Have fun playing with Winscl, but don't bother trying to use the CR (count records) command. CR makes Winscl crash every time.

## Xcacls

You've installed a domain controller (DC) with 200 user accounts. Now you have to create home directories for the users. Because you're a well-read Windows NT administrator, you know that User Manager for Domains can do much of that work for you. Your only problem is that User Manager for Domains sets a directory's permissions to full control for its user, which means you won't have access to those directories.

How can you add your user account to each directory's ACL without replacing the directory's current owner? You have two options. You can make the change in each directory, one at a time, or you can use Extended Change Access Control List (Xcacls).

Xcacls is an improved version of the NT command-line tool Cacls, which surprisingly few people know about. Xcacls lets you change the ACLs of files and directories on NTFS volumes (although it can't modify permissions on file shares). Here's the syntax for Xcacls:

```
<file/directoryname> /g <username>:<desired_file_ACLs>;[<desired_directory_ACLs>]
[/e] [/t] [/y]
```

The first parameter in Xcacls specifies the names of the files and directories whose ACLs you want to change. When you give Xcacls a file or directory name, the tool reports current permissions. For example, if I want to see the permissions on directory F1, I type

```
xcacls f1
```

This query produces the following output:

```
E:\reskit\f1 MYNTWS\Fred:(OI)(IO)F
MYNTWS\Fred:(CI)F
ORION\MarkA:(OI)(IO)F
ORION\MarkA:(CI)F
```

Two users have permissions on directory F1: Fred, whose account resides on MYNTWS, and MarkA, who has an account on the domain ORION. Each user produces two lines of Xcacls output: one for file permissions and one for directory permissions. The F at the end of each line stands for full control. Both Fred and MarkA have full control in file and directory permissions. According to Microsoft, (OI), (IO), and (CI) refer to inheritance information. I can't say I understand what they do, but in my experience, file permissions lines always begin with (OI) (IO), and directory permissions lines always begin with (CI).

The /G option in Xcacls lets you specify which permissions you want to grant a user. The /G option has three parts. The first part contains the user's name, such as ORION\MarkA, followed by a colon. The second part specifies the file permissions you want to give the user, followed by a semicolon. The third part specifies the directory permissions you want to give the user. You must

always set file permissions, but you can choose not to set directory permissions. The permissions values you can choose from are R (read and execute), C (write and delete), F (full control), P (change permissions), O (take ownership), X (execute), E (read only), W (write), and D (delete). To grant MarkA on domain ORION full control of directory F1, I enter

```
xcac ls f1 /g orion\marka:f;f
```

However, this command wipes out all previous permissions on F1.

If you want to add to file or directory permissions information without eliminating existing permissions, you can use the /E (Edit) switch. If I add /E to the end of the previous command line, Xcacls will give MarkA full control on F1 but will not delete any existing permissions on F1's ACL. If I add the /T option to the end of the command line, Xcacls will ripple the permissions change all the way down the subdirectory tree.

Suppose MarkA is an administrator who wants to add full control for himself to all the home directories located in a directory called E:\Users, without disturbing user access to those directories. He can type

```
xcac ls e:\users\*.* /g orion\marka:f;f /e
```

Now, suppose MarkA wants to kick all the users off their directories, because he is decommissioning a server. He could just leave the /E off his command, but then Xcacls would bug him with an *Are you sure?* prompt for every directory. Instead, he can use the /Y switch, which automatically answers all the prompts with Yes. His command line would look like

```
xcac ls e:\users\*.* /g orion\marka:f;f /y
```

Get to know Xcacls, and it'll come in handy any time you need to create automated backup scripts or perform home directory maintenance. Xcacls is an ACL power tool.

## Chapter 6

# Scripting Tools

### AutoExNT

Many years ago, I ran a company with two locations. Each location had a separate cc:Mail post office. The locations exchanged mail through a Lotus dial-up gateway tool, a simple DOS program that dialed in to one post office from another post office and exchanged mail between the two.

At first, I dedicated a DOS machine to the task of exchanging mail. I later tried running the gateway program on a Windows NT machine, hoping that NT would eliminate my need to run the program on a dedicated computer. The gateway worked wonderfully on NT, and it didn't require a dedicated machine. But every time the NT machine rebooted, someone needed to remember to log on to the computer and start the cc:Mail gateway. This necessity was a pain. I needed an autoexec.bat file for NT.

Now, my company uses an NT-based mail product that runs as a service. Services start automatically, so I don't need to manually start the mail exchange program every time I reboot. But I still get letters from systems administrators who need to set up non-service programs to start every time NT powers up.

The *Microsoft Windows NT Server 4.0 Resource Kit* and *Microsoft Windows NT Workstation 4.0 Resource Kit* offer the tool that I was looking for back in my cc:Mail days: AutoExNT. It lets you create an autoexnt.bat batch file and move commands to that file, just as you could do in DOS. After you create the batch file, the AutoExNT service runs autoexnt.bat whenever you power up your NT system.

### How to Make It Work

Installing AutoExNT is simple. Look in the directory in which you installed the resource kit files. Copy the autoexnt.exe and servmess.dll files from that directory into your NT machine's System32 directory. Build a batch file that contains the commands that you want to run automatically when your system boots, name the file autoexnt.bat, and put it in the System32 directory.

Next, install AutoExNT as a service. In the resource kit directory, you'll find a program called instexnt.exe. This program tells the Services database to recognize the AutoExNT service. To install the AutoExNT service, type either

```
instexnt install
```

or

```
instexnt install /interactive
```

at a command prompt. Which command you choose affects how AutoExNT works.

## Interactive AutoExNT

You must use the `/interactive` switch to run programs that need to communicate with the user. If you place a program that requires user interaction (for example, User Manager) into `autoexnt.bat` and don't use the `/interactive` switch, the program's windows won't appear. If User Manager's windows don't appear, the program is useless. You usually won't have a problem installing programs without the `/interactive` switch because most of the programs that you'll need to start automatically (such as a mail gateway) don't have an interactive component, just as NT services don't require user interaction.

If you use the `/interactive` option to install the AutoExNT service, you open yourself to interesting logistical and security questions. For example, suppose I put the `regedit` command into `autoexnt.bat`. Does the Registry Editor window appear at the initial logon screen, or only after a user logs on? Suppose the first person who logs on to the PC isn't an administrator. What kind of rights would that user have from `regedit`'s point of view? And does `regedit` appear every time someone logs on, or only when the PC first boots?

My AutoExNT experimentation demonstrated that if a program you invoke through AutoExNT isn't interactive, the program starts immediately during the machine's startup and doesn't wait for anyone to log on. However, interactive programs don't seem to do anything until someone logs on. After a user logs on, the interactive program's window appears. I'm sure NT techies have guessed the answer to the second question: Because AutoExNT runs as a service, every program AutoExNT starts has the authority of the service's account, which by default is the System Account. Placing `regedit` in an AutoExNT batch file starts an instance of `regedit` that lets any user who logs on to that machine make unlimited changes to the registry. You need to think twice about running interactive programs under the AutoExNT service. Finally, I discovered that programs in the AutoExNT batch file run only when a machine boots, not every time a new user logs on.

## Dnsserver.pl, Dnszones.pl, and Dnsrecord.pl

I sometimes need to delete large blocks of records from one of my DNS zones. You'd think that you could simply highlight a group of records in the Microsoft Management Console (MMC) DNS snap-in and press `Del`, but the DNS snap-in lets you delete only one record at a time. The foundation of any good Active Directory (AD) is a good DNS infrastructure. But unlike many other Windows 2000 tools, Win2K's DNS server isn't very controllable from the command line, so scripting common tasks is difficult. Although the `dnscmd.exe` tool ships on the Win2K Server CD-ROM, that tool isn't very complete.

However, the *Microsoft Windows 2000 Server Resource Kit Supplement One* includes a complete set of DNS command-line tools. This toolset is in two parts. One part is a Windows Management Instrumentation (WMI) object for controlling the DNS server. The second part consists of three Perl scripts that use the WMI object to control DNS servers, zones on those servers, and records in those zones. You can use those Perl scripts to tell a DNS server to do just about anything that you can tell it to do from the GUI—no obvious capability seems to be missing.

Because the tools are all scripted, you can take them apart and put them back together to do any DNS-related task—all without writing a single line of Perl. `Dnsserver.pl` (`.pl` is a common extension for Perl scripts) controls overall DNS server properties, such as starting or stopping the DNS service, clearing the server's DNS cache, and displaying the zones on the server. `Dnszones.pl` lets you work on a particular zone on a particular server. This script lets you do tasks such as cre-

ating a new zone, deleting an existing zone, or converting a zone from a standard primary zone to an AD-integrated zone. You can also use the script to enable or disable dynamic DNS (DDNS) updates on that zone and control whether the zone accepts updates from anyone and whether the updates must be Kerberos-secured. `Dnsrecord.pl` lets you operate within a zone to create, modify, or delete records. Because they're built atop WMI objects, the scripts work remotely as well as locally, so you can run a script on one computer to control a DNS server on another computer.

I'd need more space than I have here to show you the syntax for these three scripts. However, I do want to point out that you need to do a little preparation to make them work. First, install Supplement One on the computer that you'll run the Perl scripts from. That step installs the Perl interpreter and a required Perl module that tells Perl how to control WMI. But Supplement One's standard install routine doesn't install the module where Perl can find it. To install that module, create the directory `\perl\site\lib\w2rk` and copy `wmi.pm` from `\program files\resource kit` to the new directory.

Next, prepare the DNS server that you want to remotely control. You don't need to install Perl on that server, but you do need to install the WMI object that controls Perl. You'll find three files in the Supplement One CD-ROM's `\apps\dnsprovider` folder: `dnsprov.dll`, `dnsschema.mof`, and `dnsprovider.chm`. Copy those files to any directory on the DNS server (I put them in a folder I named `Dnsfiles`). Then, at a command line on the DNS server, change to the folder that you just put the DNS WMI files into and type two lines:

```
mofcomp dnsschema.mof
regsvr32 dnsprov.dll
```

You should get a confirmation after those files are installed. Then, take a peek at `dnsprovider.chm`—it's a Help file (you'll find it in the resource kit directory) that contains VBScript examples of how to use the Perl scripts.

## Ifmember and KiXtart

Write a few logon batch scripts, and you'll soon be disappointed with their shortcomings. The basic problem with batch scripts is that they're only batch files and are no more capable than batch files.

For example, suppose you're writing a batch file and you want one kind of action to occur when an administrator runs the file and another kind of action to occur when a user runs the file. More specifically, say you have a share with some administrative tools in it called `\sxl0\atools`; you want to map that share to drive `W` when an administrator logs on but not map the drive when a nonadministrative user logs on. How do you make this distinction? How do you tell a batch file that someone's in a particular group? The command interpreter, `cmd.exe`, doesn't provide one simple tool, but the *Microsoft Windows NT 4.0 Resource Kit* includes a couple of useful tools. These resource kit tools let you write more powerful batch files.

If you are an old-time NT, Windows, or DOS batch expert, you'll appreciate the resource kit's `Ifmember`. Many resource kit utilities run on a server, so you don't need to distribute them to workstations. But `ifmember.exe` is a client-side tool and needs to be present on or available to each user's workstation to work. You can put `ifmember.exe` on every user's hard disk, but that's too much work. An easier way is to put `ifmember.exe` into the same directory as the logon batch

files. (The Netlogon directory is the default directory when the logon batch file is running. Ergo, putting a program into Netlogon installs the program instantly, hands-off.)

Ifmember is a simple program; it looks like

```
ifmember group1 group2 group3...
```

where the groups are the names of user groups. If the group's name includes a space between letters, such as Domain Admins, enclose the name in quotation marks. If the person running Ifmember is a member of one of the groups named, Ifmember ends with return code 1. You can then use Errorlevel to test for this return code's occurrence, as in the following example:

```
@echo off
ifmember "domain admins"
if not errorlevel 1 goto user
echo you're an admin!
goto quit
:user
echo just a regular user
```

This batch file checks your groups to see whether you're a member of the Domain Admins group. (Note that Ifmember isn't case sensitive.) The next line, *if not errorlevel 1 goto user*, checks whether your return code is equal to 1. If the code isn't 1, the batch file skips ahead to the user: line. Then, the file displays the message *just a regular user* and ends. However, if the return code is equal to 1, the batch file runs the next line, displays *you're an admin!*, and jumps to the end of the batch file. You can implement the previous drive map example as follows:

```
@echo off
ifmember administrators "domain admins"
if not errorlevel 1 goto quit
net use w: \\sx01\atools
:quit
```

However, Ifmember works on only NT workstations. For Windows 9x machines, KiXtart is your only option. Although the official name is KiXtart 95, you can use KiXtart to build very flexible batch files on NT and Win9x workstations. KiXtart is a complete programming environment, with a comprehensive programming language using If...Then...Else constructs and Goto and Select statements, and includes nearly all the program control you find in Basic programming. KiXtart also has a rich set of built-in functions, including a function named Ingroup, which lets you write program lines such as

```
IF INGROUP("Domain Admins") RUN "net use w: \\sx01\atools"
```

Logon batch scripts might still cause you some frustration, but Ifmember and KiXtart will help you write significantly more flexible logon batch scripts.

## Logevent

Windows 2000 and Windows NT command shell scripts are great tools for automating repetitive administrative tasks. Scripts are easy to write and reliable, and you can use the AT scheduler or the Win2K Task Scheduler to schedule and run the scripts unattended. However, scripts are limited in the kinds of tasks they can perform. For instance, scripts have no built-in support for accessing the

registry or the event log. The inability to write to the event log can be particularly annoying for unattended scripts, which must record their run status information in standalone text files, which you need to manage separately from the event log.

You can get around this limitation by using the Logevent utility to tell your command shell scripts to record run status and error information in the Application event log. The Application log is a common repository for applications' status information. If you tell your scripts to write their status information to the Application log, you can have all your script and program status information in one place.

Logevent is in both the *Microsoft Windows NT Server 4.0 Resource Kit* and the *Microsoft Windows 2000 Server Resource Kit*. You can call Logevent from a batch file, and because the utility is a standalone executable binary file, it's convenient to use. You don't need to do anything special to install it—it doesn't need any registry entries or DLLs. Just make sure that the command shell script can find the logevent.exe file.

Logevent's syntax is straightforward:

```
Logevent [-m <\\ComputerName>] [-s ] [-c ] [-r "" ] [-e ] [-t ] ""
```

You use the -m option to specify the computer that will log the events. If you omit this switch, Logevent uses the local system. The -s option specifies the severity level of the events you want to record: You can choose S (success), I (information, the default value), W (warning), E (error), or F (failure). Use the -c option to specify the category number, which you can use to filter events in the NT Event Viewer. If you don't specify a value, Logevent assigns the category 0 (None). The -r option lets you supply a string (e.g., the application or script name) that identifies the source of the logged event. Logevent uses User Event if you omit this option. The -e option lets you assign an event ID, which is useful for filtering entries in the event log. You can use the -t option to set a timeout value for trying to write a log entry. By default, Logevent waits 60,000 seconds before giving up and exiting. The last parameter is a required string that Logevent will write as the event log's description. If you type only "logevent" to invoke the utility, Logevent displays a usage prompt.

Let's look at how to use Logevent. If I want to build a command shell script that needs to log its success or failure in the Application log, I would construct a script like this:

```
@ECHO OFF
:: The first parameter must be GO
@if NOT "%1"=="GO" GOTO :LOGERROR

:: Write the success event
logevent -s S -r "Script Event"
"Script completed."
GOTO :EOF

:LOGERROR
:: Log the error
logevent -s E -c 3 -r "Script Event" -e 99 "Script Failed! Invalid parameter"
::EOF
```

This sample script tests its first parameter for the word GO. When the script finds a match, Logevent writes a success entry to the Application log that records the source as Script Event and the description as *Script completed*. Otherwise, the script executes the lines that follow the

:LOGERROR tag and writes an error entry with a source of Script Event, an event ID of 99, and the description *Script Failed! Invalid parameter*.

Although Windows Script Host (WSH) and VBScript are Microsoft's preferred scripting technologies, nothing works quite like a batch file for doing things quickly and reliably. Logevent.exe is a useful utility for administrators to have in their tool belt. For more information about using the Logevent utility, see the Microsoft article "How to Use Logevent.exe to Log Events from a Batch File" (<http://support.microsoft.com/?kbid=131008>) or Chapter 14 of the Win2K resource kit.

## Munge

Imagine that you recently completed a huge task—you consolidated several servers. A monster server named saturn that has RAID, clustering, and so on now contains the shares and applications formerly on atlas, redstone, and titan. Just as you finish the job, a chilling thought occurs to you: the logon scripts. You have hundreds of logon scripts, each with hard-wired references to atlas, redstone, and titan.

You pull up one of the scripts and take a look. The script contains lines such as Net Use v: \\atlas\pix and \\redstone\desktopapp\vsfan c: d: e:. When you moved the shares to saturn, you retained all their original names. So the solution is simple: Just edit all the logon scripts and change all the atlas, redstone, and titan references to saturn. Although this task is simple to state, completing it would be torture. However, computers are good at this kind of repetitive task. And munge.exe is the tool you need to automate such a task.

Munge is a *Microsoft Windows NT Server 4.0 Resource Kit* command-line search-and-replace tool. To use the tool, create an ASCII script file that lists the changes you want to make. Then, feed Munge the script file and the file you want to change. Munge makes the changes you request and writes the changes to a file with the original file's name. To be safe, the tool keeps the original data file but changes its extension to .bak.

Suppose you have a file called animals, and you want to change every instance of cat in the file to dog. You need to use Notepad or another ASCII text editor to create a script file (e.g., called changes) with the line

```
cat dog
```

Then, invoke Munge by entering

```
munge changes animals
```

Munge reports all the changes it made, then exits. The animals file then contains those changes, and you have a file on your hard disk named animals.bak that contains the contents of the animals file before you ran Munge. (In case you're wondering, if you ask Munge to perform search-and-replace operations on files with the extension .bak, the tool complies but stumbles over a bug: Munge changes your modified file's extension to .mge and erases the original .bak file.)

If all your logon batch files were in one directory, you could use Munge with an old batch trick, the For...Do command, to complete the search-and-replace operations. First, create the script file (e.g., called srvchg) with the lines

```
atlas saturn
redstone saturn
titan saturn
```

for the three change commands. Then, rather than using Munge on each file (i.e., munge srvchg filename), use the For...Do command as follows to tell NT to complete the search and replace on a group of files:

```
for %f in (*.bat) do munge srvchg %f
```

For each file with the .bat extension in the current directory, the statement finds the file, constructs the command munge srvchg %f (substituting the file's name for %f), and tells NT to execute the command. Then, the For...Do statement repeats the cycle for the next .bat file in the directory. The For...Do statement finally stops when it runs out of .bat files.

Munge has a few quirks. For example, if your script file contains nonalphanumeric values (e.g., \\atlas \\saturn), Munge requires that you surround each item with quotes. Thus, you'd type

```
"\\atlas" \\saturn
```

This requirement creates problems with case. You want the search and replace to work whether the server name is atlas, ATLAS, or something in between. When you specify before and after values without using quotes, Munge is case insensitive. But when you surround the before value with quotes, Munge is case sensitive. In this case, the tool would change only \\atlas (not \\ATLAS) to \\saturn.

Munge's online Help discusses several options that I couldn't get to work. And unfortunately, this Help is the only documentation available. Although Munge is fairly buggy, it's a handy tool if you use it for the kind of tasks in my examples.

## Service.vbs

Centralizing and automating any Windows NT shop requires some way to reach across a network to determine which services are running on a given server, examine how those services are configured (e.g., do they start up automatically or manually, do they have a service account associated with them), and change the configuration remotely. You can buy products to help with these tasks. However, NT's Server Manager can also do many of these jobs. Server Manager replicates the Control Panel Services dialog box and can display that dialog box for any machine in its list of servers. But you can't control Server Manager from the command line, so it can't help you if you need to use a batch file to turn services on or off. Of course, you can use the Net Start and Net Stop commands to turn a service on and off, but only on the local computer.

The *Microsoft Windows NT Server 4.0 Resource Kit Supplement 4* includes a script named service.vbs that lets you start or stop a service remotely. To shut down the DHCP server service on a remote machine named \charlie and restart the service, you could execute the commands

```
cscript service.vbs /x /n dhcpserver /s charlie
cscript service.vbs /g /n dhcpserver /s charlie
```

These examples assume that \charlie recognizes you as an administrator. If not, your command can include the name and password of an administrator account. You use the /u option to specify the user account and the /w option to specify the password. If \charlie recognizes an administra-

tive account named `adminguy` with password `swordfish`, you can start the DHCP server on `\charlie` with the command

```
cscript service.vbs /g /n dhcpserver /s charlie /u adminguy /w swordfish
```

Many other `service.vbs` options exist. The following command contains a short list (you can use the `cscript service.vbs /?` command to get the complete list):

```
cscript service.vbs <actionoption> /n <servicename> /s <servername> /u
<adminaccount> /w <password>
```

*Actionoption* can be `/g` to start a service, `/x` to stop a service, `/r` to remove a service from the computer, or `/d` to list a service's dependencies. To find out a service's name, you can use the `/l` (i.e., list) option. For example, if I type

```
cscript service.vbs /l /s charlie
```

I'll see a list of all the services on `\charlie` (whether they're running or not), their display name (e.g., Microsoft DHCP Server), and their actual name (e.g., `dhcpserver`). Again, you might need to use the `/u` and `/w` options to specify a username and password for an administrator account.

Nifty as `service.vbs` is, it can't do every service-related task. I don't see a way to use `service.vbs` to change the name of the service account for services that don't use the standard "system" account. However, you can use the script to remotely install a service. First, you copy the file (or files) that makes up a service to the remote machine. For example, you can use the Net Use command to access an administrative share and copy the file to `\winnt\system32`. Then, you just type

```
cscript service.vbs /i /e <filename> /n <servicename> /c <displayname>
```

For *filename*, substitute the full pathname to the service's program file. Replace *servicename* with the service's short internal name (e.g., `dhcpserver`), and replace *displayname* with the display name (e.g., Microsoft DHCP Server). The command to install a new service that controls that wireless automatic lawnmower of yours might look like

```
cscript service.vbs /i /e c:\winnt\system32\buzz.exe /n mowrservice /c
"Ronco Automatic Lawnmower Service"
```

Remember that perhaps the best thing about `service.vbs` is that, because it's a script, you have the source code. So, you can use it as a springboard to craft an even better utility.

## VBScript Tools in Supplement 4

The *Microsoft Windows NT Server 4.0 Resource Kit Supplement 4* includes a wealth of new administrative tools written not in the traditional way (i.e., as C code resulting in opaque `.exe` files) but in VBScript. Because VBScript is an interpreted language, the programs are visible for your inspection with Notepad. (Hmm, could this fact mean that Microsoft is embracing the open-source movement? Nah, that thought is too frightening.) These scripts exhibit a commendable level of consistency, and Supplement 4 provides a lot of them—I counted 62 in my copy of Supplement 4. Let's look at what some of these scripts can do.

One group of tools—`ps.vbs`, `pstop.vbs`, `eventlogmon.vbs`, and `logmeminfo.vbs`—monitors the system's state. You can run these tools on the local system, or you can use the `/s` option (case is irrelevant) to run them on a remote system. As Task Manager does, `ps.vbs` lists the processes that are running on the system, but it provides an extra benefit: `Ps.vbs` also shows you where the programs' executables reside. For example, in addition to telling you that `lsass.exe` is running and has process ID 136, `ps.vbs` reveals that the program that is `lsass.exe` is in `C:\winnt\system32\lsass.exe`. `Ps.vbs` puts all this information into the standard output device `Stdout`, of course, so you could write another script to do something with that process information. The command

```
ps /s somemachine
```

will show you the processes running on a machine named `somemachine` (note that you don't need to prefix the name with two backslashes).

As `ps.vbs` does, `pstop.vbs` lists process names and IDs, but it lists them in decreasing order of the amount of CPU time that each process has used. (Even better, `pstop.vbs` doesn't bother to report that "idle" has taken up the lion's share of the time.) `Pstop.vbs` can also use a `/s` option to report on a remote machine. If the remote machine doesn't recognize your user account and you have an account that the machine does recognize, you can offer that account's name and password with the `/u` and `/w` options, respectively. For example, the command

```
pstop /s bigpc /u ignatz /w swordfish
```

uses an account named `ignatz` and the password `swordfish` to interrogate a machine named `Bigpc`.

`Eventlogmon.vbs` is a neat script that lets you watch items appear in the event log as applications add the items. After you run `eventlogmon.vbs` in a command window, the utility continues to run until you press `Ctrl+C`. `Eventlogmon.vbs` lets you do tasks such as keep an eye on the Security log of a computer that you suspect an intruder is trying to access. Because `eventlogmon.vbs` runs on remote systems as well as the local computer, monitoring for an intruder is simple.

The last tool in this group, `logmeminfo.vbs`, is also the simplest. `Logmeminfo.vbs` displays the current status of memory—physical, virtual, pagefile, and available memory.

Another group of Supplement 4 scripts lets you inventory characteristics of machines around the network. All these tools are similar and give you information about the hardware that each tool's name suggests: `keyboard.vbs`, `listdisplayconfig.vbs`, `listprinters.vbs`, `motherboard.vbs`, `parallelport.vbs`, `pointdev.vbs`, `processor.vbs`, `scsicontroller.vbs`, `serialport.vbs`, `sounddevice.vbs`, and `tapedrive.vbs` (I'm sure I've missed one or two).

The real value of these scripts probably doesn't lie in what they can do right now. Supplement 4 includes the source code for the scripts, so they provide actual working examples of the power of VBScript. Do you need to do inventories around your network? Imagine the cool script you could build by cutting and pasting pieces of the inventory scripts. You could build a custom script that collects from the network detailed hardware information about your company's computers and peripherals, then dumps that data into a database. With a bit of extra and snazzier coding, the script could feed the data to a Microsoft Access database—sort of a poor person's Microsoft Systems Management Server (SMS).

## WinAT, Soon, and Sleep

Three tools in the *Microsoft Windows NT Server Resource Kit* and *Microsoft Windows NT Workstation Resource Kit* can make the Windows NT Schedule service's AT tool even better. I use the Command Scheduler (`winat.exe`) more than any other resource kit tool. If you've used the AT command, you know that it can be frustrating to work with. You must type long strings of commands, and if you mistype one character, you have to start over. The Command Scheduler lets you use a GUI to implement the Schedule service.

The Command Scheduler lists the jobs in a computer's schedule queue, each job's ID, the day and time each job is scheduled for, and whether the job includes the `/interactive` option (which lets an AT program display dialog boxes and receive commands via the GUI). Double-click an item in the schedule queue, and the Command Scheduler provides a dialog box that lets you adjust the job's features. Like the Schedule service, the Command Scheduler can control schedules for every NT machine on your network.

I also frequently use the resource kit's `Soon` tool. Suppose I want to run a program called `beep.bat` that makes my computer beep. I have to look at my computer's clock and schedule `beep.bat` for a later time. If the computer thinks the time is 11:43 A.M. and 37 seconds, I might open a command line and type

```
at 11:44 c:\beep.bat
```

If I press Enter before the clock hits 11:44, the computer will beep. However, if I'm not quick enough, the computer will schedule the command for 11:44 A.M. tomorrow.

The `Soon` tool eliminates this timing problem. It lets you schedule events for a future time, relative to the current time (in seconds). For instance, you can use `Soon` to schedule a beep for 1 minute from now, rather than for 11:44. `Soon` replaces `AT`, and the syntax is similar to `AT`'s:

```
soon <\\machinename> <delay_in_seconds> [/interactive] <command>
```

`Soon`'s greatest strength is scheduling repetitive commands. For example, suppose I want my computer to beep every hour. I can write the following batch file, which I'll call `beepit.bat`:

```
@echo off
echo ^G
soon 3600 c:\beepit.bat
```

I created the `^G` in the second line of this batch file by holding down the Ctrl key and pressing the G key. Because I did not include a machine name, the computer I typed the command on will beep. To start the beeping, I type

```
beepit
```

from a command line. The system will beep once, and `Soon` will schedule the batch file to run again an hour later.

The third AT-related resource kit tool, `sleep.exe`, lets you schedule a program to wait for a specified period (also in seconds) before continuing. Suppose you want to schedule a backup of a computer's D drive, and you need to kick all users off the server. First, use the Net Pause Server command to keep users from attaching to the server. Next, use the Net Send `*` command to notify

users who are already on the server that their connection will terminate in 5 minutes. Use the Sleep command to instruct the computer to wait 5 minutes before continuing. Finally, use the Net Stop Server command to kick everyone off the server. Here's the command sequence:

```
@echo off
net pause server
net send * Please get off this server. It will go down in 5 minutes.
sleep 300
net stop server
ntbackup d: ... <whatever_options_you_need>
net start server
```

The Command Scheduler, Soon, and Sleep are three powerful resource kit tools that I use regularly to control the NT Schedule service. Now, if only they could control my daily schedule....