1

Administering Vista Security: The Little Surprises

Much of this book shows you how Vista's new big security technologies work, how they'll affect you, and where you can control them. This chapter, however, doesn't hit the big stuff; instead, in this chapter I want to introduce you to a bunch of changes in Vista that are fairly significant, but not obvious…until you run up against the kind of strange, unexpected, or puzzling behavior that I've come to think of as "Vysteries." Now, you might think "hey, if this is a potpourri of small Vista administration and security surprises, why not put it at the *end* of the book?" I thought about that but realized that if you *did* want to fire up a copy of Vista and work through some of the things we cover in the rest of the book, then you might find yourself more aggravated from tripping over the small brambles at your feet than from trying to scale the high towers of User Account Control internals and the like—so the first chapter seemed a practical place to put these items.

But let me stress that these aren't all *bad* surprises. All I'm trying to do in this chapter is give you a quick heads-up about things that I feel have changed most significantly administration-wise, particularly from a security point of view, with a view to highlighting the not-so-well-publicized changes. That way, you can decide best where to spend your time in Windows new doodads. (In addition, I'm hoping to show you these things before some client mentions it in a meeting. Don't you just hate *those* kinds of surprises?) These aren't in any particular order; it is, again, a potpourri.

Because, as I mentioned in the Introduction, I'm trying to keep this short and because I'm working from pre-release versions of Vista, I'll assume that you've already figured out how to get Vista up and running in at least a minimal manner on a test system or two. That way, we can move right along to the surprises.

Restoring the Administrator

You go to log onto Vista for the first time, and want to log on as the Administrator, just as you always have. But there's this hitch because, well, there doesn't seem to *be* an Administrator account anymore. Arrgh.

Actually, the Administrator account's still there and can be logged onto. It's just disabled. So here's how to get it back.

First, log onto the Vista system as a local administrator. If you're on a domain, that means that you'll probably need to log on with a domain administrator account, or, if you're not in charge of your domain, then ask your domain administrator to put your domain account in the Administrators group of your Vista machine. If you're using a computer that's a member of a domain, but you *can't* do either of those things then you're probably stuck, unless you reinstall the Vista box as a member of a workgroup rather than a domain.

Making Your Own Administrator

If, on the other hand, you're running a Vista box that is *not* a member of a domain, then Vista will prompt you to create a user account when it first starts up. Vista then automatically puts that account in the Administrators group, just as XP did. It won't *force* you to give that account a password, but it's a good idea to do it anyway because Vista, like XP and 2003, treats accounts with blank passwords as sort of second-class citizens in that they can't be used over a network.

Because that first account is a local administrator, you may not actually need to revivify the Administrator account.

Activating the Administrator Account

Do you, then, need to activate the Administrator account? Probably not. I figured out how to activate the Administrator account in the early days of Vista, but soon realized that I could accomplish anything with that account that Vista prompted me to create that I could do with the Administrator account. In fact, when testing Vista builds 5472, 5536, and RC1 I never even bothered with activating the Administrator account.

I have heard of people needing the Administrator for application compatibility; as some folks have apps coded to run using the Administrator account (not a good idea, but, again, I've been told that some need it). In any case, if you need the Administrator back, then here's the sequence. First, the Administrator account needs a password, as it's currently blank and, as we all know, having an account on a system named "Administrator" with a blank password and that is a member of the Administrators group is a terribly bad idea.

Also, if your system is a member of a domain that has minimum password requirements installed, then you won't be able to activate an Administrator account with a blank password. (Not that the error message that you get from Windows is crystal clear in explaining why it errors out when you try to activate an Administrator account with a lame password; you tell it to activate the Administrator account and it replies something to the effect that "the password does not meet the minimum requirements of this system." You then scratch your head and say, "I wasn't *trying* to do anything with a password!")

We'll give the Administrator a good password and activate it at the same time. Here's how.

Note that in my instructions, I'm using the "Classic Start menu." You'll see that I also run using the Windows Classic theme, which leads to my Vista desktops looking sort of like Windows 2000. I do that mainly for the sake of better speed and quicker response time.

1. Log onto your Vista system with whatever local administrator account you've wangled.
2. Start up a command prompt: click the Start button (it doesn't say "Start" anymore, but it's in the same place as the old Start button, the lower left-hand corner by default and is a circular representation of the Windows flag). Then click All Programs, and then Accessories.
3. I know, I've lulled you into a false sense of "I know what I'm doing now," and you're about to click the Command Prompt icon. *Don't.* Instead, right-click the Command Prompt icon and choose "Run as administrator." You will see your desktop go gray and you'll see a dialog box warning you that you're about to do something administrator-like, and did you really mean to do that? You then click either a Continue or Cancel button.
4. This is called the "Consent user interface" because the program that kicks it off is called `consent.exe`. It's part of User Account Control (UAC), which we'll discuss in Chapter 2. You'll see this dialog box every time you do something that requires even mildly "administrator-ness" to work right. It stays up for two minutes, and if you don't respond in those two minutes, you get a dialog box announcing that Windows won't run the program because "The operation returned because the timeout period expired." In any case, click Continue to get Vista to open a command prompt.
5. Now that you've got the command prompt, set the Administrator's password to something other than blank. (And, if necessary, something that makes your domain's group policies happy.) That command looks like net user administrator *newpassword*. In my case, I'll type **`net user administrator swordfish`** to give it the password "swordfish." As with virtually all Windows command-line commands, case does not matter except in the password itself, and you've got to press the Enter key once done. You should get "The command completed successfully."

But what if you didn't? If you get "System error 5 has occurred. Access is denied," then you didn't start up the command prompt by right-clicking and choosing "Run as administrator." Yes, I know, you're logged on as an administrator, you should be able to do administrator things…but it's a longer story having to do with UAC, and we'll cover it later. For now, just please remember to always start your command prompts with "Run as administrator" if you want to do anything administrative.

6. Now we've got an administrator with a good password; finish the job and activate the account. From the command prompt, type **`net user administrator /active:yes`** and press Enter.

I did that as two commands for clarity's sake, but you *can* do it in one: **`net user administrator swordfish /active:yes`** will work as well.

And no matter which path you took, be sure to clear your screen or prying eyes might see that new password. In fact, closing the command prompt window at that point might be a good idea so that no one can press the Up arrow to see what you typed.

Power Users Are Essentially Gone

The Power Users group has always been an attempt by Microsoft to provide a group that wasn't quite as powerful as the Administrators group, but more powerful than the Users group. That need, as you probably know, grew out of the fact that the permissions and rights that you get from being a member of the Users group just aren't sufficient to allow you to run many applications. Many of us administrators know that we need to wean users away from having local administration accounts, but need to give them more power so that they can run applications. For some folks, the Power Users group was the answer.

Unfortunately, it wasn't a very good answer. Because Power Users have the ability to write to `ntoskrnl.exe`, the basic Windows program, then an evil (and, of necessity, very smart) Power User could modify that file, giving themselves more powers and escalating their privileges. They also have the ability to modify at least one system service that runs as the LocalSystem account, which would let a crafty Power User log onto the system as the LocalSystem account. (See Mark Russinovich's blog entry at `http://www.sysinternals.com/blog/2006/05/power-in-power-users.html` for a more detailed write-up of how these attacks would be launched.)

Power Users was created as a Band-Aid to help solve the overall problem of application compatibility because many applications won't work when run from a so-called standard user account.

Jargon alert: "standard user" is a relatively new Microsoft phrase that you'll hear more and more as you start using Vista. It means "a user who's basically just a member of the local Users group on a machine," with no special administrative powers at all. You'll see this appear in phrases like "once you have all of your users running as standard users...."

The idea with Power Users was to create a group that had just enough administrative powers to allow users to run those troublesome "gotta be an admin to run me" applications. In other words, Microsoft created Power Users to work around the fact that many Windows developers did a lousy job of testing their applications. (And, unfortunately, some of those developers work for Microsoft.) That has led to a world where millions of users are logged into their systems all day as administrators or power users, with the result that the millions of machines that they're working on are prone to security problems.

The alternative solution to the problem of applications that require administrator-level power—chivvying developers into writing software that works properly when run by a standard user—seemed too highly priced before we all starting facing the worms and spyware du jour, but Microsoft has apparently decided that the time has come to ask those developers to pitch in and help solve the security problem.

As a result, Microsoft has essentially eliminated the Power Users group. It's still present in case you've got some resource that has a permission that refers to the Power Users group, but it's "Power" Users in name only, as it basically has the same power as the Users group. To see this change, try creating a member of the Power Users group on an XP system and then use the `whoami.exe` application found in various versions of Support Tools and the Resource Kit (or built into the OS, in the case of XP x64 and Vista) to find out what privileges that user has. Run `whoami /priv` and you'll get this list:

- `SeChangeNotifyPrivilege`
- `SeShutdownPrivilege`
- `SeUndockPrivilege`
- `SeSystemtimePrivilege`
- `SeProfileSingleProcessPrivilege`
- `SeCreateGlobalPrivilege`

Do the same thing with a Power User member on a Vista system, open up a command prompt and run `whoami /priv` and you'll see this:

- `SeChangeNotifyPrivilege`
- `SeShutdownPrivilege`
- `SeUndockPrivilege`
- `SeTimeZonePrivilege`
- `SeIncreaseWorkingSetPrivilege`

(If you're wondering what all of that Se*something* stuff means, we'll cover it the next chapter.) Let's make that a bit more understandable by looking first at the things that old and new Power Users have in common and which things they see differently. Both groups have `SeChangeNotifyPrivilege`, which you may know better as "bypass traverse checking." It means that if a user has NTFS permissions to access a particular folder, but the folder is nested inside folders that the user is denied access, then the user can get to the folder that they *do* have permission to. (Users have had this permission since NT 3.1.) The other

two rights that both XP and Vista Power Users have are `SeShutdownPrivilege` and `SeUndockPrivilege`, which mean pretty much what they sound like: the power to shut a system down or to undock a laptop from a docking station.

Both XP and Vista users have the right to modify system time, but in different ways. XP Power Users could change time, date, and time zone. Vista users and Power Users have been granted an altogether new right: the ability to change just the time zone, something that Microsoft added in response to customer demand. You see, regular users don't have the right to change workstation times because Microsoft's always seen that as a security issue. (Certainly it might make Kerberos unhappy in a domain, as domain authentication breaks down if a client's clock is more than five minutes different from a domain controller's clock.) But, people argued, they had regular old users who traveled with laptops and crossed time zones. Admins wanted the itinerant laptop users to be able to change their clock's time zone, but not the time. So Vista has the new "change time zone" right, and Power Users (and Users) have it, but Power Users no longer have the right to change workstation time.

XP Power Users had two rights that Vista Power Users don't have: "profile single process" (`SeProfileSingleProcessPrivilege`) and "create global objects" (`SeCreateGlobalPrivilege`). The first was scary because it allows one user to examine a process being run by another user. The idea is this: suppose I wanted to run Performance Monitor and monitor processes that other users are running. That would require the OS to give me the right to gather statistical information about other users' processes. The power is normally used to allow someone to run a program like Performance Monitor to watch things that the system is doing but in theory could be used to spy on someone or even on the system. (There's an even more intrusive one called `SeDebugPrivilege`, but we'll meet that next chapter.)

The second is a right that controls whether or not users can create something called "file mapping objects in the global namespace." And no, that wasn't an English sentence, but I'll translate. When programs need to read or write disk files and folders, or talk to and from other programs, or control an I/O device like a printer or a serial port, there are often several ways to do that, coding-wise. But one way that's been around for a long time is to treat all of those kinds of I/O as being particular kinds of disk file reads and writes; so, for example, to let one program talk to another program, a programmer could do a lot of complicated coding, *or* she could have the two programs that she wants to communicate create a sort of imaginary file. Then, when the two programs needed to talk between themselves, they just read or write from or to that imaginary file. Where does security come in here? Well, if I'm just a regular old user and I want to run program A, which talks to program B, then those programs—which logged on as me, because I started them—will need the ability to create these imaginary files, these "file mapping objects." Clearly even the lowliest of users should be able to create one of these; no security worries there. Ah, but now let's add in Terminal Services, and we get a problem. More than one user may have a TS session running on the same system, and each user may run programs that use these file mapping objects. What if they collide, what if User 1 runs a program that creates a file mapping object named "Wally," and User 2 runs a program that tries to do the same thing? That'd be a problem, except for the fact that every user session gets its

own separate area or "namespace." Each user in a Terminal Server session, then, has his own "local namespace" in which he can create file mapping objects. But Terminal Services has a way for one user's session to interact with another user's session, through a *global* namespace for file mapping objects. Being able to monkey around with the *global* namespace could enable a bad guy to affect other TS users' sessions, so Microsoft removed that ability from Power Users in Vista. Offhand I do not know of any applications that this change would break if you take this right from a user, but it's possible. In that case, just add the right to a particular group and put the user in that group because, again, Vista Power Users (and standard users) lack `SeProfileSingleProcessPrivilege` and `SeCreateGlobalPrivilege`.

The Vista Power User has one right that the XP Power User didn't—the right to bump up the amount of memory that a given application uses, the "working set" of that app. Apparently Microsoft needed to create that specific new right for some reason and needed regular users to have it.

Once you've seen the privileges and group memberships of a Vista Power User, try creating a normal user account in Vista and run `whoami /priv` on it for comparison: you'll see that members of the Users group have the exact same rights as members of Vista's Power Users group.

So Power Users are much less powerful rights-wise. How about Power Users' NTFS permissions, though? Yup, they've been circumscribed. Even my up-to-date, SP2-equipped copy of XP x64 edition that I'm writing this on gives vast powers over the System32 folder to Power Users, vast when compared to what little a member of Users can do…but there's nary a special-to-Power-User permission on *any* folder of a Vista system that I can find, much less a scary one!

"Run..." Is Off the Start Menu

I'm not sure who makes the call on the user interface stuff at Microsoft, but I get the impression that he thinks we user types are pretty dumb. It seems like every version of Windows changes the default behaviors of the Start menu in ever-increasing levels of annoyance. XP hid Administrative Tools, Server 2003 made getting to the actual Control Panel more work, and now Vista has taken away the Run… item from the Start menu. Personally, I use Run… a lot of the time, if for no other reason than to quickly get to Regedit and the local Group Policy Editor. Losing Run… on the Start menu makes me less productive on Vista.

So let's fix that, shall we?

1. Right-click the Start menu, and in the resulting context menu choose "Properties." As with earlier versions of Windows, that brings up the Taskbar and Start Menu Properties page with the "Start Menu" tab highlighted. Choose the "Customize…" button in the upper right-hand corner to bring up the Customize Start Menu dialog.
2. In the Customize Start Menu dialog, you'll see a number of radio buttons in a window with a scroll bar down its side. Scroll down almost to the bottom, and you'll see the option "Run command" with an unchecked check box next to it. Check the box.

3. Still in that window, scroll all the way down and you'll see a section called "System administrative tools." In that section, choose the radio button labeled "Display on the All Programs menu."
4. Click OK to dismiss the Customize Start Menu dialog, and again to dismiss the Taskbar and Start Menu Properties page.

The Run… command and the Administrative Tools group are now back, after a few clicks. (And they call Vista a productivity tool!)

BOOT.INI Is Gone, BCD Is Here

Now and then, I need to edit the `boot.ini` file in order to fix some configuration issue. Ever since NT 3.1, it's been an ASCII text file on the hard disk.

With Vista, that's all changed; it maintains a boot file called the Boot Configuration Data or BCD located on the boot volume (that is, the volume that the operating system boots from, no matter what Microsoft calls it) in a folder named BOOT. It's one of those files locked open by the operating system (like the *.EVT event log files), so you can't edit it in the normal manner, *and* because that means that it'll be tougher for the odd bit of malware to modify it.

Don't go looking to edit it from the Control Panel, either; the Startup and Recovery dialog box is still in Control Panel hidden a few layers down, but where the XP version of that dialog had a button labeled "To edit the startup options manually, press Edit," that doesn't exist in Vista anymore. Instead, there's `bcdedit.exe`, a command-line tool for messing with Vista boot options.

boot.ini Review

The reason why *I* needed to modify `boot.ini`—normally a few-minute operation that became a multi-hour process, although it'll take you much less time after reading this—is that when I'm running test machines that are not connected to the Internet, either virtual or real, I'm often using *slower* machines, and in an effort to reduce my waiting time when playing with Vista, I like to turn off Data Execution Prevention (DEP). I do *not* recommend doing this on a production machine or, for that matter, any system into which you will type any data that you wouldn't want the world to know. But for test systems that you won't be sharing your vital data with, it's a great idea. With XP and 2003 systems, I could always shut off DEP by editing the `boot.ini` and adding the `/NoExecute=AlwaysOff` option to any `boot.ini` entry. But how to do that (and other things) to BCD? Well, to learn that, we've got to learn BCD-ese. Here's the `boot.ini` on my XP workstation:

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(2)\WINDOWS
```

```
[operating systems]
multi(0)disk(0)rdisk(0)partition(2)\WINDOWS="XP x64 " /fastdetect /
NoExecute=OptOut
multi(0)disk(0)rdisk(0)partition(2)\WINDOWS="XP x64 w/debug" /fastdetect /
NoExecute=OptOut /DEBUG
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional"
/fastdetect
```

This particular `boot.ini` offers three different operating system options when booting this computer; those three options are in the section named `[operating systems]`. The three lines following it (each is long and broken on the page, but there would indeed be just three lines if we were viewing this on a wide computer screen) is called a "`boot.ini` entry." For example, consider this one:

```
multi(0)disk(0)rdisk(0)partition(2)\WINDOWS="XP x64 " /fastdetect /
NoExecute=OptOut
```

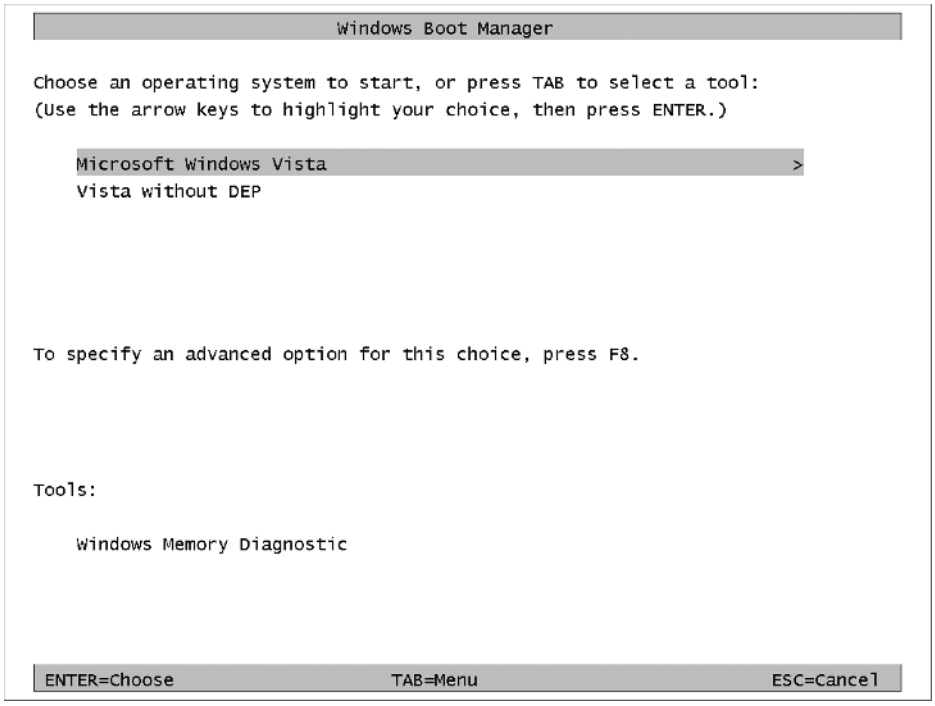
The `multi(0)disk(0)partition(2)\WINDOWS` is just an arcane way of saying "the actual operating system is on the second partition of the first hard disk, and in the Windows directory on that partition." That's followed by two "switches," `/fastdetect` (which tells Windows not to bother looking around for devices attached to parallel and serial devices, which hasn't been generally necessary since 2000 came out), and `/NoExecute=OptOut`, which is the normal setting for DEP. Because there are three operating system entries, I see a `boot .ini` menu offering those three every time I boot my workstation. Other useful switches are `/maxmem`, which tell your copy of Windows to not use your system's RAM above some level, or `/debug`, which enables system debugging, or `/numprocs`, which tells your system to ignore some number of processors.

Above the `[operating systems]` section, there is a `[boot loader]` section. It specifies two things: how long to leave the menu on the screen, and which option to make default if the `boot.ini` options time out.

Now, if you're scratching your head saying, "I never see anything like that `boot.ini` file, or a boot-time menu at all in either XP *or* in Vista," that means that you've got only one operating system entry. In that case, you don't get the menu on either XP or Vista. If you *do* have a Vista BCD with more than one entry, then you see a different boot menu from the one that you would have in the pre-Vista days, assuming that you had a multi-entry `boot.ini`. The Vista boot menu is text, but it's a bit snazzier than `boot.ini`, like the one that you see in Figure 1.1.

This menu shows two options: "Microsoft Windows Vista," the option built when Vista's installed, and "Vista without DEP," an option that I've created and that I'll show you how to create. In addition to the operating system entries, Vista's Boot Manager also offers the option of booting straight to a memory tester—a convenient touch on Microsoft's part, particularly given that Vista systems typically need quite a bit more memory than XP systems.

FIGURE 1.1 A Vista system with multiple boot options



BCD Terminology

To work with BCD, we need to learn a bit of BCD-ese. What we might think of as the entire BCD "database" is called the "store" or the "system BCD store." The store contains one or more "entries," which act as `boot.ini` entries did; thus, were I to translate my `boot.ini` into a BCD, I'd have a *store* containing three *entries*. There is, in addition to the entries, a tools menu that by default contains just one entry, the memory tester. Each entry may contain what we used to call `boot.ini` switches, like `/NoExecute=AlwaysOff`, but they're not called "switches," they're called "entry options."

Let's see how to relate this to an actual BCD by telling bcdedit to dump the current configuration. Do that by opening a command prompt as an administrator (right-click the Command Prompt icon, choose Run as administrator, and confirm the choice when UAC asks), and then type just **`bcdedit`**. I get an output like this (I've shortened a few items for clarity):

```
C:\Users\mark>bcdedit
```

```
Windows Boot Manager
--------------------
identifier         {bootmgr}
device             partition=D:
description        Windows Boot Manager
locale             en-US
inherit            {globalsettings}
default            {current}
displayorder       {current}
                   {c0e803c8-217c-11db-8f12-0016364dab15}
toolsdisplayorder  {memdiag}
timeout            30

Windows Boot Loader
-------------------
identifier         {current}
device             partition=C:
path               \Windows\system32\winload.exe
description        Microsoft Windows Vista
locale             en-US
inherit            {bootloadersettings}
osdevice           partition=C:
systemroot         \Windows
nx                 OptOut

Windows Boot Loader
-------------------
identifier         {c0e803c8-217c-11db-8f12-0016364dab15}
device             partition=C:
path               \Windows\system32\winload.exe
description        Vista without DEP
locale             en-US
inherit            {bootloadersettings}
osdevice           partition=C:
systemroot         \Windows
nx                 AlwaysOff
```

Notice that you see three sections in this report: a "`Windows Boot Manager`" section and two "`Windows Boot Loader`" sections. Remember the `[boot loader]` section? It has morphed into the Windows Boot Manager information. Each entry in the `[operating systems]` section gets its own Windows Boot Loader section.

Creating a Second OS Entry

Let's start putting bcdedit through its paces but…safety first! When installed, Vista creates one OS entry called "`Microsoft Windows Vista`." If you think that you'd like to play around with changing boot options then I highly recommend it, if for no other reason than to take advantage of my suggestion about speeding up test machines with that DEP configuration notion that I've already mentioned. But instead of mucking with the one boot entry that you've got, I even *more* highly recommend that you first make a second OS entry and do your experiments on *that* entry. After all, it *is* possible to make your system unable to boot with a bad OS entry, and that is guaranteed to ruin your whole day. (Unless you like watching Vista install. I mean, it *does* have that lovely "undersea view of the bottom of a kelp forest" background while installing.)

How to create a second OS entry? That's one of bcdedit's abilities. The easiest way to create a second OS entry is to just copy the existing one with the `bcdedit /copy {`*`ID-of-entry-to-copy-from`*`} /d `*`description`* command. I will explain *{ID-of-entry-to-copy-from}* in just a couple of paragraphs but for now we can use `{default}`, which is the identifier for the default operating system entry. Using that information, I originally created my "Vista without DEP" OS entry like this:

```
bcdedit /copy {default} /d "Vista without DEP"
```

When I did that, I got a response of

```
The entry was successfully copied to {c0e803c8-217c-11db-8f12-0016364dab15}
```

I'm going to explain that thing in the curly braces—it's called a globally unique identifier or GUID—next, but before I do, let me just summarize where we are at this point. If you try that command on a Vista system and reboot, you will get to see the Windows Boot Manager and your new "Vista without DEP" entry that, at the moment, doesn't do anything different than the "Microsoft Windows Vista" entry. *But* now you've got a safe OS boot entry to play with.

Understanding Vista Boot Manager Identifiers

What's with those `{default}` and `{c0e803c8-217c-11db-8f12-0016364dab15}` things? Windows Boot Manager needs some way to be able to identify the multiple operating system entries. Now, it *could* give them names like "default Vista OS entry," but that would be, um…okay, I don't know why they don't let you just give them arbitrary identifiers; it just seems to be something that's been in Windows since Windows 2000. The idea is, I suppose, that you might go crazy and accidentally create *two* OS entries with identifiers of "default Vista OS entry," and then your computer would implode. Anyway, when Vista creates a new OS entry, it also generates a random 128-bit number and uses that as the OS entry's "true name." Now, *inside* that OS entry is something called a "description" and you and I can fill it with text like "Vista without DEP" or the like, and you and I will use that to identify a particular OS entry, but Vista just sees that "Vista without DEP" name not as a *real* name, but instead as window dressing—`{c0e803c8-217c-11db-8f12-0016364dab15}` is the true name for our new "Vista without DEP" OS entry as far as software's concerned.

That means that when you want bcdedit to do something to a particular OS entry, then you'll usually have to identify the entry that you want to configure. Usually that'll be the GUID of the OS entry. But you will sometimes be able to save a little work, as GUIDs aren't the *only* kind of OS entry identifier that bcdedit will take. It also recognizes the `{default}` and `{current}` identifiers. Note that they're surrounded by curly braces, as are the GUIDs. `{default}` is an identifier that tells bcdedit, "I want you to configure that OS entry that starts up by default, but I don't want to look up its GUID." `{current}` does the same thing, but it identifies the OS entry that the system is currently booted into. Thus, if you're working on a Vista system that booted into the default operating system entry, then both `{default}` and `{current}` point to that OS entry.

So, back a page or two, when I offered the command `bcdedit /copy {default}...`, I was telling `bcdedit` to copy whichever operating system entry was the one I'd get by default. When bcdedit spat back the big number in the curly braces, it was telling me that GUID of the OS entry that it had just created for me.

If you ever need to *see* the GUIDs of your computer's default OS entry, just type **`bcdedit /v`** and you'll get the same long listing as you saw a few pages back when I typed just "bcdedit," *except* that instead of seeing `{current}` on the Identifier line, you'll get the GUID of that entry. Both a GUID surrounded by curly braces or the predefined `{current}` or `{default}` items are called "identifiers" by bcdedit.

Choosing Timeout and Default OS with bcdedit

Now that we're experts on identifying OS entries, let's return to some nuts and bolts. As with `boot.ini`, Windows Boot Manager's main jobs are to define a timeout value and a default. (Clearly there are also other things that Windows Boot Manager does, but I'm trying to cover just the essentials here.)

Changing the Boot Manager Timeout

To change the timeout value, type **`bcdedit /timeout`** ***`numberofseconds`*** to set the number of seconds that Windows Boot Manager waits before choosing the default operating system entry. For example, to tell Windows Boot Manager to wait 15 seconds, you'd type

```
bcdedit /timeout 15
```

The adjustment you'll want to do more often is probably choosing the default operating system instance.

Changing the Default Boot Manager Entry

You'd think the second task—telling Boot Manager which OS entry to load by default—would be a snap. It is, *almost*; you can pick any OS entry and make it the default, but, as you'd probably guess by now, you've got to refer to that OS entry by an identifier, and the chances are good that you'll have to use its GUID.

As we've already seen, the new "Vista without DEP" OS entry on my system got a GUID of `{c0e803c8-217c-11db-8f12-0016364dab15}`.

Even if you type into your system exactly the same commands that I've typed, you will not get the same GUID, as they're random. So if your GUIDs look different than mine, don't panic, it's supposed to work out that way.

Using that GUID, I can then make that entry the default by typing **`bcdedit /default {`*`guid`*`}`**, so for example to make "Vista without DEP" the default, I'd type

```
bcdedit /default {c0e803c8-217c-11db-8f12-0016364dab15}
```

Again, you can do something similar on your system; just remember that you'll have to retrieve the particular GUID of your "Vista without DEP" OS entry; simply typing **`bcdedit`** by itself will, recall, show you your OS entries and their GUIDs. And don't forget to surround the GUID with curly braces; bcdedit won't work without them. Then, after typing **`bcdedit`** all by itself a second time, I'll see the same output, except in the "identifier" line the `{c0e803c8-217c-11db-8f12-0016364dab15}` will be replaced by `{default}`. The other OS entry, the "Microsoft Windows Vista" one, will have an identifier of `{current}`.

Changing an Entry Option

With our new OS entry created and set to the default, we're ready to start with playing with entry options. Recall that "entry option" is the bcdedit phrase for what we used to call "`boot.ini` switches." Some switches have values, like the `/NoExecute=AlwaysOff` example that I've already offered, and some, like `/basevideo` (which says to boot the system with the basic VGA driver) don't have values, and you enable them by including them in the OS entry and disable them by leaving them out. In BCD and bcdedit, however, every entry option has both a name, like `NoExecute`, *and* a value, like AlwaysOff. (Case seems not to matter to BCD and `bcdedit`, in my experience.) `Boot.ini` switches that didn't previously have a value, like "/basevideo," now get a value of "yes" or "no." (`/basevideo` is now called simply "vga," by the way.)

You can include an entry option by typing **`bcdedit /set [{`*`entry guid`*`}] `*`entry-option-name`*` [`*`entry-option-value`*`]`**. To set `nx` to `AlwaysOff` in the currently running operating system entry, then, we could type

```
bcdedit /set {current} nx AlwaysOff
```

If, however, we hadn't included an OS entry at all, then bcdedit would have assumed that we wanted that change done on the currently booted OS entry anyway, and so this would get the same job done:

```
bcdedit /set nx AlwaysOff
```

To set `nx` in the default OS entry, we'd type

```
bcdedit /set {default} nx AlwaysOff
```

To tell the OS entry with a GUID of `{c0e803c8-217c-11db-8f12-0016364dab15}` to boot using the standard VGA video driver, we could type

```
bcdedit /set {c0e803c8-217c-11db-8f12-0016364dab15} vga yes
```

(Just to be clear, that command would be typed as one line.)

Now that you know how to modify boot options, here are a few of the available Vista boot options in BCD and, for the `boot.ini` black belts out there, the corresponding `boot.ini` switches of each option entry:

`nx`, as I've mentioned, controls DEP. Its `boot.ini` value was just `/NoExecute`. `nx` can be can be set to `AlwaysOn`, which applies DEP to all user applications and operating system programs; `AlwaysOff`, which does not apply DEP to anything; `OptOut`, which applies DEP to everything except particular programs that you exclude; or `OptIn`, which applies DEP to all operating system programs and any applications that you add in. (You can do the excluding or including in the Control Panel's System applet.)

`vga` is, as I've already explained, the setting telling your system to forgo whatever video driver it's currently using and instead use the generic VGA driver. It takes values "yes" or "no." Its `boot.ini` counterpart was `/basevideo`.

`numproc`, which lets you limit your OS to a certain number of processors, was also `/numproc` in `boot.ini`, and takes a number; `bcdedit /set numproc 1` would tell your system to only run one processor on the currently running OS entry. This can be useful because once in a while, you'll run into an application that was only tested on single-processor systems but that contains bugs that only pop up in multiprocessor computers.

`removememory` lets you exclude some amount of memory from Vista. Its `boot.ini` counterpart was called `/burnmemory`. It takes a value in either decimal or hex (prefix a hex number with "0x" so it recognizes it as hex) of the exact number of bytes of memory to give Vista—specifying "500000" would remove about a half a megabyte of memory from Vista, not about a half a gigabyte!

`truncatememory` is another command restricting the amount of RAM that you allow Vista to use. Where removememory specifies how much RAM to take away from Vista, leaving it the rest, truncatememory specifies how much RAM to *give* to Vista, denying it the rest. You wouldn't use both of these in the same OS entry, by the way. As with removememory, truncatememory takes a number as a parameter. That number is, like removememory, the exact number of bytes to give Vista. Truncatememory's name in `boot.ini` was `/maxmem`.

If that's still not clear, imagine that you've got a system with 2 GB of RAM. removememory 500000000 would remove a half gig, leaving 1.5 GB of RAM for Vista. You could do the same thing with truncatememory, but you'd feed truncatememory 1500000000.

`quietboot` skips the GUI's little animated rectangles that ripple left-to-right as an indicator that the OS is loading. Set it to "yes" or "no." It was `/noguiboot` in `boot.ini`.

`sos`, which was named `/sos` in `boot.ini`, tells the operating system to show each driver and service's name as the operating system boots. This can be useful if your system locks up on boot; just sos to "yes" and reboot (clearly you need a different way to boot to make this setting, perhaps another OS entry!), and the name of the last driver loaded may be the culprit. This takes "yes" or "no" for parameters.

bootlog tells your system to create a log of the drivers that the OS loads, in the order in which it loads them. It then saves that log in Windows directory in a file called `ntbtlog.txt`. This option takes "yes" or "no" and was called `/bootlog` in `boot.ini`.

Cleaning Up: Deleting OS Entries

That's about all I wanted to cover in BCD and bcdedit to help you tweak your OS's starting parameters. But if you find that you've got your OS entry just the way you like it, and don't need the one automatically built by Vista, then you might want to tidy up a bit. You can delete an OS entry with the `bcdedit /delete` *`identifier`* command. For example, on my system, I'd first type "bcdedit" to find out the GUID of the now-unused OS entry, discover that it was `{24a500f3-12ea-11db-a536-b7db70c06ac2}`, and type

```
bcdedit /delete {24a500f3-12ea-11db-a536-b7db70c06ac2}
```

"Documents and Settings" Is Gone, Kind Of

I liked Windows 2000's improvements over NT 4.0, but I really found one thing annoying about it: the Documents and Settings folder. I do a lot of command-line work, you see, and folder names with spaces are a pain in the neck. You've got to put quotes around them, and even if you do, some programs get a bit stupid when handed a folder name with spaces in it.

Vista, however, does make working with folders and file names with spaces in them easier. Whenever you're using a command-line tool that requires a file or folder name, you can just type as much or as little as you like of the folder or file name that you want to specify, then press the Tab key. It auto-completes the file or folder name. Thus, to change my directory to `C:\Documents and Settings`, I just type `cd d` and then press Tab, and instantly the command becomes `cd 'c:\documents and settings.'` If there is more than one directory starting with a "D" and it chose the wrong one, I'd just press Tab again and it'll cycle through the possibilities. It even puts the quotes around the name if there's a space in the name. (This feature existed in 2000, XP, and 2003 but was not enabled by default.)

NT originally stored user profiles in `winnt\profiles`, but Microsoft decided to move the profiles out of the OS's directory (which probably made sense) into a separate location. That, again, was a good idea; calling it "Documents and Settings," in contrast, was a bad one. (Not as dumb as making people learn goofy phrases like `HKEY_LOCAL_MACHINE` to understand the Registry, but dumb enough.) Vista changes that, creating a folder to store local profiles called `\Users`. You've just gotta love it: no spaces, short and sweet.

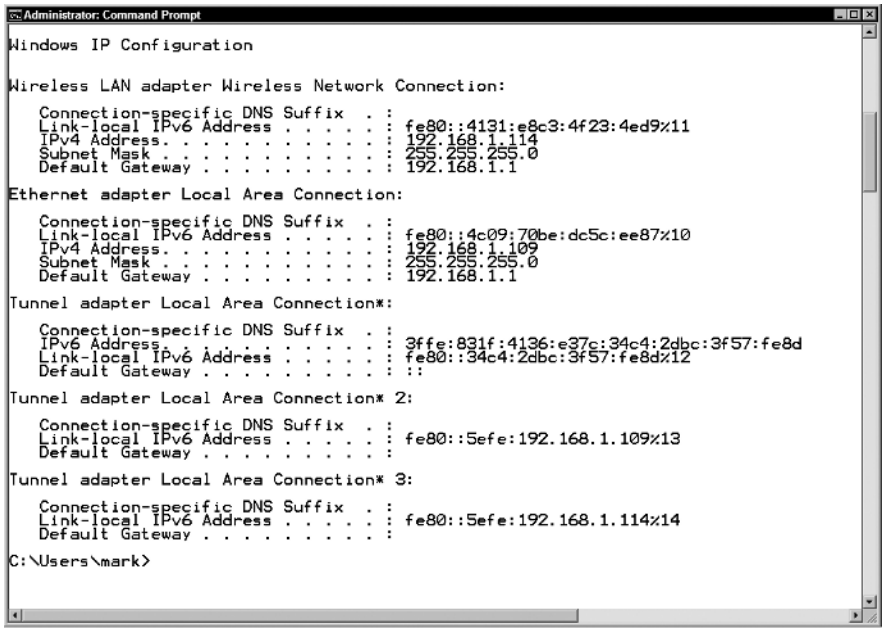
But we've been living with Documents and Settings for six and some years, so Microsoft knows that there will be *some* application out there that doesn't follow the rules, and decides to write some data to `c:\documents and settings\`*`some-users-name`*`\`*`some-folder-name`* instead of just asking the operating system where that user's profile folder is. To combat that, Microsoft creates a Documents and Settings folder on the drive, *but hides it*. Then they take things a step further and set its NTFS permissions to—you'll love this—*deny the Everyone group read access to Documents and Settings*. The result? Any application that tries to create data in Documents and Settings, rather than just asking the OS where to put the profile information, will fail.

What should you do if you happen to find that one bad application out of thousands? Either get the developer to fix the problem, or unhide the folder and change its NTFS permissions. Or Vista's file and folder virtualization feature may fix the problem invisibly. (You'll learn more about that in Chapter 3.)

IPv6 and Network Properties

The first time I started an early copy of Vista (back when it was known as "Longhorn Desktop"), it appeared that my network connections weren't working. So I did the same thing that most network admins would do: I opened up a command prompt and typed **`ipconfig`**, then pressed Enter. Figure 1.2 shows something like what I saw.

FIGURE 1.2 `ipconfig`, with a new, scary look



First of all, this particular computer's only got two network adapters, not…umm, give me a second, let me count…*five* network adapters. Auugh. What's *going on* here, I wondered? "Why are there colons and percent signs where there are supposed to be periods, and…oh, I see. IPv6." As you may know, the version of IP that we use in TCP/IP for the Internet is actually "IPv4," IP version 4. There was an IPv5 because some folks played around with a networking tool to make transmitting multimedia stuff easier—I don't think it went anywhere—and meanwhile, while the world started worrying about running out of IP addresses, some smart folks started working on a new version of IP based on 128-bit, rather than 32-bit, addressing. Were the Internet to go IPv6, we'd *never* have to worry about running out of addresses, as a little calculation shows that we'd have 429,446,885,032,372,986,132 IP addresses for every square inch of the Earth's surface and, yes, that *does* include the part of the earth covered by water.

Now, I understand why Microsoft wanted to put IPv6 into Vista. It's the basis of a lot of interesting technologies in mobile computing; in theory, when the whole world goes IPv6 then you can get up in the morning, grab your e-mail on your notebook via the wireless network in your house, take a bus downtown and catch up on the news at your favorite websites while your laptop's connected to the municipal wireless LAN, and then finally connect to the network at work to hook up with your domain…and never change an IP address. Microsoft's committed to taking big market shares in the mobility world, so I can see that might be a reason why they installed IPv6 by default, whether I want it or not. China's working to produce a country-wide IPv6-based Internet, and China's an important sales market, so I can see why Microsoft decided to install IPv6 by default, whether I wanted it or not. Microsoft hates hearing that they're Johnny-come-latelies to any network party and most Linux distributions don't install IPv6 by default yet, so I could see that Microsoft might like using IPv6 as a chance to seem more "wave of the future" than their biggest desktop rival.

But boy, are they gonna pay for it, at least in my opinion. Microsoft hates running a tech support operation because it's not nearly as profitable as is cranking out Vista DVDs and charging a few C-notes apiece, but the need's there; turn off tech support for Windows and a lot of users will find something else.

I think, however, that Microsoft's gotten something of a free ride tech-support-wise over the years. As I've already noted, I'll bet that everyone reading this book has probably acted as a no-charge-for-services tech support person for friends, family, and neighbors. Now imagine what happens the first time one of the legions of free support folks start trying to troubleshoot a neighbor's cable modem troubles with the neighbor's brand spanking new copy of Vista. One look at *that* ipconfig output, and many of those helpful volunteers will just say "um, maybe you'd better call the cable company."

And you know how helpful the cable guys will be when they see those IPv6 numbers.

Now, let's be clear, and this is a very important point: as far as I can see, home network users can keep IPv6 running without any ill effects to their network at all. But I've run into problems with IPv6 on XP, 2003, and Vista systems in an Active Directory environment—dynamic DNS registrations run into some trouble, presumably because the IPv6 addresses were confusing, and the problem persists into RC1—and so I'd prefer to have to *add* IPv6 rather than remove it. But inasmuch as it's the default, that means that I'll have to remove it

if I want my ipconfig outputs to look less cryptic. Now, you're probably thinking, "that'll be easy; just go to the Properties page of your network interface card and uncheck the boxes next to IPv6." My thoughts exactly the first time I ran Vista.

But then I went *looking* for the Properties page of my NIC. Want to find it? Well, then put on your mining helmet with the lamp on it; we're going into the bowels of Vista....

1. Click the Start button, then choose Control Panel. You'll see that on Figure 1.3, although I've added an annotation.

   This isn't a gratuitous screen shot, because I need it to illustrate a point about Control Panel. Look at the section that I outlined with a black rectangle; it's the area where you kick off network-related Control Panel pages. But here's the surprise: that's not just one section, as was the Network and Internet Connections section in XP's Control Panel. No, it's not obvious, but those are actually pointers to three separate sections: "Network and Internet," "View network status and tasks," and "Share files and folders." Each is a separate hyperlink-like thing, and each takes you to a different place. None of them, however, will take you directly to your NIC or NICs.

2. Instead, click "View network status and tasks," and, in the resulting Control Panel page, you'll see a list on the left-hand side of that page labeled "Tasks." One of those tasks is "Manage network connections;" click that to see something like Figure 1.4.

FIGURE 1.3 Vista's Control Panel

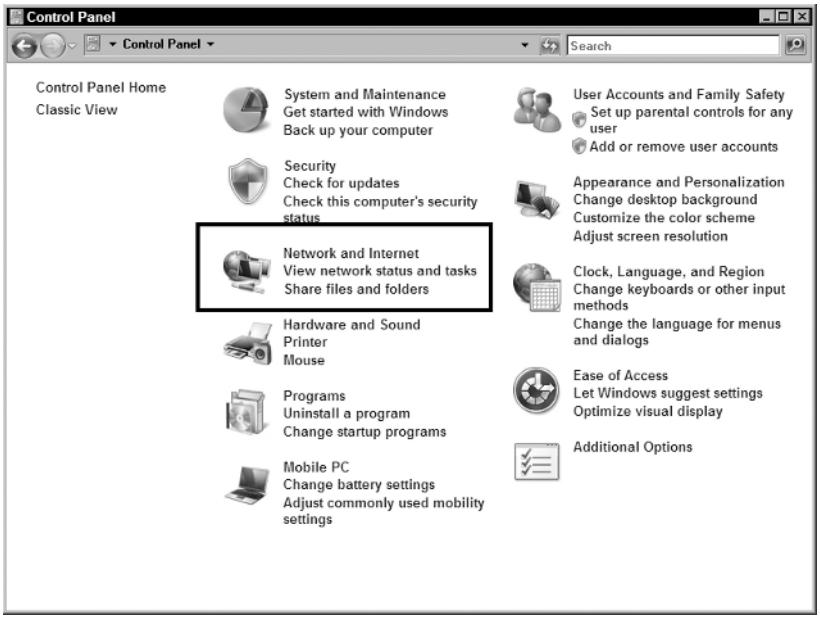
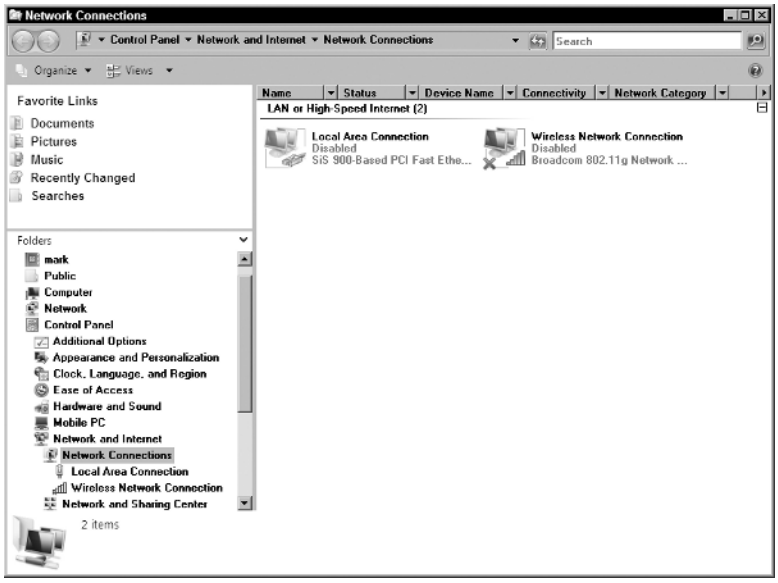
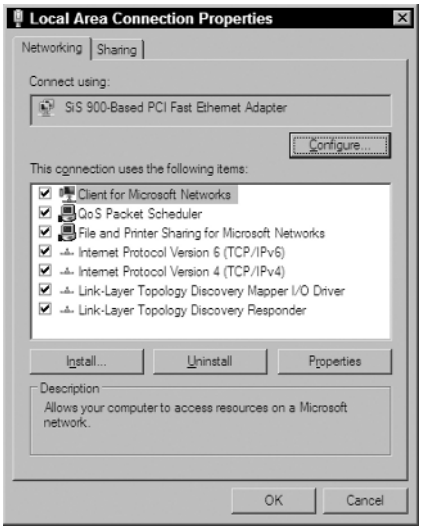


FIGURE 1.4 Network Connections, Vista style



3. Now, *that* seems a bit more familiar. Ignore the fact that both NICs are designated "disabled;" that's an artifact of build 5472, the build that I took this from, and appeared fixed at RC1. Right-clicking the Ethernet NIC (yours will, of course, almost certainly be a different make or model, but the dialog will look similar) and choosing Properties shows something that looks like Figure 1.5.

FIGURE 1.5 Vista's NIC Properties page



4. Once again, a more familiar dialog box…almost. Note all of the IPv6 and IPv6-to-IPv4 compatibility tools. Uncheck any that you want, and some of the IPv6 stuff goes away; three references to "tunnel adapters" remain.

Now you've seen how to find the NIC Properties page. I should stress that I'm not instructing people in general to get rid of IPv6; I just wanted to show you how to find the NIC Properties page, and IPv6 was a good excuse.

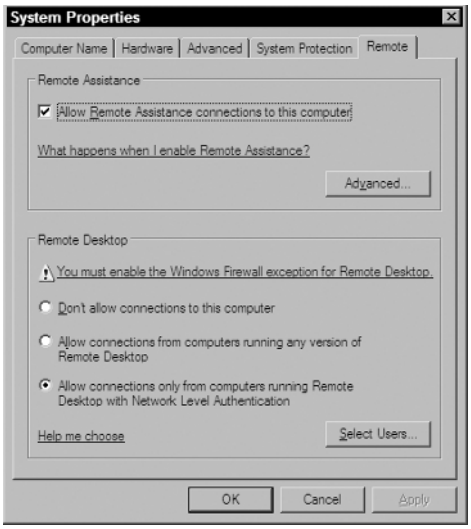
Remote Desktop Gets a Bit More Secure

At first glance, Remote Desktop for Vista looks pretty much identical to RD on XP. But a slightly closer look shows a small but important change in security. You can see this in the Remote tab of the System property page. Get to it like so:

1. Click the Start button.
2. In the resulting menu, right-click Computer and choose Properties.
3. In the Control Panel page that appears, look at the Tasks list on the left-hand side of the page. Choose "Remote settings." You'll see a property page like Figure 1.6.

As I said, this looks similar to the corresponding page in XP, but notice that instead of two options—"enable or disable remote desktop"—there is a third offering, "Allow connections only from computers running Remote Desktop with Network Level Authentication."

FIGURE 1.6 Configuring Remote Desktop in Vista



To understand this, think about what's happened every time you've tried to use Remote Desktop to remote into a system. You start up the Remote Desktop Connection (RDC) app in XP or 2003 and tell the app to connect you to some system. RDC goes out and, assuming that Remote Desktop's enabled for that system and they've got their firewall set up so that people can remote in, you get a logon screen from the remote system. Now, from the point of view of a particularly paranoid security person, this is interesting: you haven't authenticated to this system yet, but it's responded to your command for its attention nonetheless. In other words, Remote Desktop is a little bit more trusting than it could be, as the sequence of events is (1) request a Remote Desktop connection from the remote system, (2) the remote system stops what it's doing and creates a remote session to your computer, and (3) you log on.

By choosing the new third setting under Remote Desktop, you tell Remote Desktop to switch steps (2) and (3). When you try to log onto a remote system that supports this approach, which Microsoft calls "Network Level Authentication," you don't see a remote standard Windows logon dialog sitting atop a remote desktop; instead, you get a dialog box like the one in Figure 1.7.

But does this mean that a Network Level Authentication logon only works against Vista systems at the moment? Apparently yes. As I write this in September 2006, Microsoft has released a package called "Remote Desktop Connection 6.0" for XP SP2, 2003 SP1, and the x64 versions of XP and 2003. They did not release it to the general public, and it was only available from Microsoft's beta software site, but I'd be surprised if it weren't either generally available with Vista's release, or might even end up on the Vista DVD. But even with this updated RDP client, you cannot do a Network Level Authentication against a Vista system or, if you can, I've not figured out how.

What if you still want older systems to be able to remote into your system, but you'd like any Vista systems trying to log in to use Network Level Authentication? Then choose the second radio button. Vista clients will still use Network Level Authentication even if the Vista system they're remoting into doesn't require it. Is it a bad idea to enable the second radio button? Well, of course. On the one hand, enabling it means that you can RD into your Vista box from a wider variety of clients; on the other hand, the whole point of Network Level Authentication was to lessen the chance that someone could tie up your computer's CPU with bogus attempts at Remote Desktop sessions, and the second radio button leaves open that possibility. Once again, security and compatibility are sometimes tradeoffs.

FIGURE 1.7 A Network Level Authentication logon dialog

Oh, hey, I almost forgot my favorite new Remote Desktop feature. You can cut and paste files across a Remote Desktop connection. Want to deliver a folder from your desktop to the computer that you're remoting into? Just right-click it, choose Copy, and then left-click on some folder in the remote system, right-click, and choose Paste. Quite nice, although as far as I can see, the revised RDP client for XP and 2003 doesn't support this. The revised RDP client *looks* as if it'll manage that drag and drop, but when you drop, nothing happens.

NTFS and the Registry Are Transaction Based

This falls in the category of a good surprise, in fact a really nice one. Both the file system and the Registry are now transaction based in Vista. This surprised me because it was supposed to appear in Server 2007 but it's in Vista. "Transaction based" means that you can take a number of separate files, copy, move, or whatever operations you need, and essentially package them up so that they're all or nothing. If one of the operations fails, then you just "roll back" and everything done so far is undone. Here's an actual example run:

```
Microsoft Windows [Version 6.0.5456]
(C) Copyright 1985-2005 Microsoft Corp.

C:\Users\mark>transaction /start
A transaction has been successfully started.
Transaction ID: {1288b5a4-4b58-4006-88d8-6bc86f4b8ad3}

C:\Users\mark>md newfiles

C:\Users\mark>copy con newfiles\test
hi there
^Z
1 file(s) copied.

C:\Users\mark>dir newfiles
Volume in drive C has no label.
Volume Serial Number is 4834-858C

Directory of C:\Users\mark\newfiles

07/17/2006 06:48 PM <DIR> .
07/17/2006 06:48 PM <DIR> ..
07/17/2006 06:48 PM 10 test
1 File(s) 10 bytes
2 Dir(s) 15,731,507,200 bytes free
```

```
C:\Users\mark>transaction /rollback
The current transaction has been rolled back.

C:\Users\mark>dir newfiles
Volume in drive C has no label.
Volume Serial Number is 4834-858C

Directory of C:\Users\mark

File Not Found

C:\Users\mark>
```

Here, I start a transaction, then create a new folder and put a file in that folder. But then I cancel the transaction, and it's all undone; asking for a directory listing of the new folder yields "File Not Found." In contrast, typing `transaction /commit` would have said "transaction's over, make it all permanent." Where will this be useful? Well, File and Registry-based transactions will be pretty useful for applying patches. Heck, you could actually install and test a piece of software, and then uninstall it via a transaction rollback. But that'd only work if the software didn't require a reboot; any reboots act as a `transaction /rollback`. I suspect we'll find plenty of pretty valuable uses for this. (I've got to say it again: the word "patches" keeps coming to mind.)

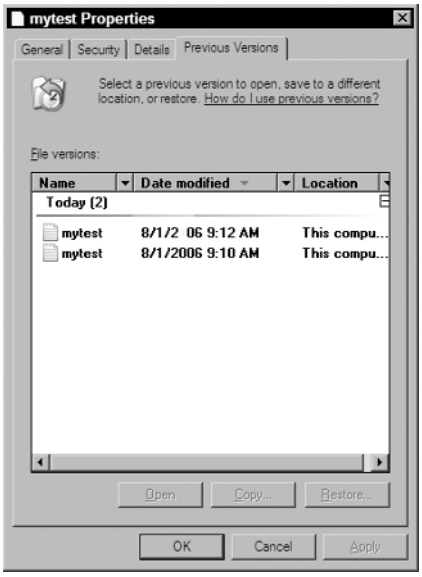
Unfortunately around RC1, Microsoft took the `transaction` command out of Vista. Apparently the under-the-hood support for transaction-based NTFS and Registry is still there, but the command itself posed some theoretical problems and so Microsoft decided that letting regular users like you and me set up transactions would be a bad idea. So unless they change their minds, then transactions will be something that only programmers can set up. (Which might make sense; it's just a shame.)

Undelete Comes to Windows for Real!

If you've ever used System Restore for XP, then you know how useful it can be. System Restore takes periodic snapshots of the state of your operating system and lets you roll back to before you installed the Driver from Hell or that antivirus application that seems to work by crashing your system, which is of course one way to keep you from getting malware, although not the optimal one. Now, with Vista, System Restore does the same thing for your *files*. Right-click any file or folder and choose Properties, and you'll see a dialog box like Figure 1.8.

Note the tab named Previous Versions. That's right, version*s* with an "s." Decided that your version of the Great American Novel was better two days ago and you didn't back up? No worries; check out Previous Versions and just grab the version from a couple of days ago. As before, you get a System Restore point once a day by default, or whenever you tell System Restore to create one.

FIGURE 1.8 Previous Versions tab in a file's Properties page



Changes in Security Options

If you've got any security-related responsibilities are all, chances are good that at some point you've opened up the Group Policy Editor either on the local group policy object or on a domain group policy object and navigated to Computer Configuration ➢ Windows Settings ➢ Security Settings ➢ Local Policies ➢ Security Options. That folder contains a grab bag of options that let you dial your security up or down. Turn 'em all off, and you're basically running a NT network, circa 1993 security-wise: loose and easily hacked, but compatible with ancient software and operating systems. Turn 'em all on and you'll need to be running only XP, 2003, and Vista systems and new applications.

But most of these security settings are just on/off ones, and they've *got* to have some default, out-of-the-box value. Microsoft's got to pick those defaults, and it's a tough job. They've been slowly tightening security not just with every version of Windows but often with every service pack, so it should be no surprise that Vista not only continues to offer the same security options as XP SP2 did, but more options, as well, *and* tighter settings on some existing options. In this section, I'll enumerate which security options settings have changed, and how that may affect your job of fitting Vista systems into your existing network.

But don't panic: yes, Vista tightens some security settings that have been left loose for a long time, but remember, these are settings in the *local* group policy object and can be easily rolled back if necessary. Furthermore, if you're connected to a domain, and that domain has a domain-based group policy object that specifies values in the Security Options folder, then those settings will override the local settings on a Vista system. The only real downside you might experience if you monkey with the security settings a lot would be a little heartburn from Vista when you try to initially join a Vista box to an existing Active Directory domain, but I'll discuss that later.

A note before we get started: when I say that something has "changed from XP" or "changed from Server 2003," I am referring to the 32-bit versions of XP and 2003. I mention this because as I look at the default settings in the local group policy object of my x64 workstation I note that some of the "new Vista settings" are actually already included in the x64 versions of XP and Server 2003.

Changes to Named Pipe Access

Named pipes are a way for programs to communicate among themselves. Years ago, most named pipes created by the OS were poorly secured or not secured at all. Many hackers successfully attacked Windows systems through poorly secured named pipes. One of the easiest avenues for these sort of attacks was by connecting as an "anonymous" user. This is a once-obscure but sadly now well-known way to connect to many Microsoft protocols and, as its name suggests, you needn't use a username and password to log in; you can, instead, remain anonymous. While allowing anonymous users *any* access to a Microsoft network resource isn't a very good idea, the fact is that for backward compatibility purposes Windows still uses some anonymous connections. Microsoft's been slowly removing the anonymous user—if I recall right, the first code to reduce the power of "anonymous" was as far back as 1998 with NT 4.0 SP3—but it's still around, and Vista takes up cudgels to reduce its power—and threat—a bit further, in these changes in how named pipes handle anonymous users.

Windows has, since XP at least, had a Security Options setting "Network access: Named Pipes that can be accessed anonymously." It lists a subset of the system's named pipes that you need the anonymous user to be able to access, and by default the group policy setting has included a bunch of stuff that doesn't really make sense:

- COMNAP and COMNODE only appear on a server running Microsoft's gateway software for talking to an IBM mainframe, their "Host Integration Server" (HIS). To the best of my knowledge, it's not possible to run HIS on any of Microsoft's desktop OSes.

- SQL\QUERY would appear on a system running Microsoft SQL Server or its equivalent. It's *possible* that a Vista system might be running SQL Server Express 2005—although not possible, I am told, to run its predecessor, Microsoft Desktop Engine (MSDE)— but not likely.
- LLSRPC appears only on servers running the Licensing Service. Why Microsoft would want anonymous people accessing the Licensing Service is a puzzle, and in no case would it appear on a desktop OS.
- BROWSER allows a system to act as either a master browser or backup browser on a subnet; this pipe is how the master and backup browsers talk. If the backup and master browser on a given subnet are not members of the same forest, then they need to be able to anonymously access the BROWSER named pipe so that the master browser can send the backup browser a copy of the segment's browse list. Microsoft has kept BROWSER in the list of named pipes that can be accessed anonymously because of that case where a workgroup might have backup and master browsers. In any case, the chances that it's a desktop OS are small, but not impossible; one could imagine a small home workgroup built entirely of Vista systems. But in that case we'd probably be talking about a single segment, where broadcasts could handle any name resolution needs.

Vista's default setting for "Network access: Named Pipes that can be accessed anonymously" removes all of those named pipes, leaving just one: SPOOLSS. That works with the print spooler, and *that's* a server role that is quite common for desktop OSes.

Why did Microsoft have so many silly named pipes in XP's default set of group policy settings? They were just saving themselves a little trouble by creating a set of defaults that they could apply both to server OSes and workstation OSes. With Vista, it looks as though that's no longer true, and the desktop has gotten its own set of settings.

Changes to Share and Registry Access

Criminals wanting to penetrate a system anonymously have avenues other than named pipes. Windows systems also feature a variety of built-in shares, and some can be accessed anonymously—or could be, before Vista. By default, XP allows two shares to be accessed anonymously. They are named in a Security Option setting named "Network access: shares that can be accessed anonymously." The two named on XP are COMCFG, which is related to the Host Integration System, and DFS$, which is necessary for a server hosting a Distributed File System root. Since Vista can't host HIS *or* a DFS root, they're both gone in Vista, and "Network access: shares that can be accessed anonymously" features an empty box.

Another avenue for bad guys is to read parts of the Registry remotely, as it reveals inner details about the system. Some Registry paths clearly must be accessible over a network for networking and remote administration to function, and those are named in the Security Option setting "Network access: remotely accessible registry paths." Where it names nine paths in XP, there are only three in Vista.

LM Deemphasized, NTLMv2 Emphasized

Vista changes two items in Security Options in a way that I personally like a lot, but I want you to understand what they might do to your network compatibility. The two settings are

- "Network security: Do not store LAN Manager hash value on next password change" is now enabled, even though it had been disabled by default until now.
- "Network security: LAN Manager authentication level" changes from "Send LM and NTLM" to "Send NTLMv2 response only."

First, a little background. It's nice that we've got machines called domain controllers (DCs) that centrally store usernames and passwords so that every machine doesn't have to carry around a complete list of all usernames and passwords. But having central logon services means that you've got to figure out how to *use* those services safely. I say that because if I'm sitting a Workstation A and offer it my domain name and password, then how's Workstation A going to ask Domain Controller B to verify it? It's not a very good idea to ask across the network "hey, Domain Controller B, I've got a guy here who says he's a guy named Mark with password 'swordfish,' does that sound right?" It'd be a bad idea because people can easily listen in on network traffic.

So, instead of making DCs and workstations reveal secrets across the wire, Microsoft and other vendors do something a bit more crafty. When the workstation says to the DC, "I want to log Mark in, can you verify that this is indeed Mark talking to me," the DC replies "okay, *I* know Mark's password, and you've got the password that the would-be Mark offers. So here's a big random number. Take it and encrypt it, using the password that this might-be-Mark guy offered. Then send the encrypted result back to me." The workstation takes the random number (which is called the *challenge*), encrypts it with offered password, and sends that encrypted number (which is called the *response*) to the DC. Meanwhile, the DC also encrypts the challenge, using the password that it has on file for Mark. If the response from the workstation matches the encrypted challenge, then the DC knows that the password typed in to the workstation is indeed the correct one, without the workstation having to transmit the proffered password.

Now, that's just the broad outline of how this kind of authentication method, called a "challenge-response mechanism," works. The devil's in the details: complex encryption's hard to crack (and that's good) but requires more CPU power, long keys are harder to crack but also require more CPU power and make logons slower (and that's bad), so the OS vendor has to weigh its choice of encryption methods and key lengths carefully. As time has gone on, Microsoft has created three different challenge-response mechanisms: the late 1980s offered "LM," short for "LAN Manager," which was replaced in the early 1990s with "NTLM," the NT version of the LAN Manager logon protocol, and in the late 1990s Microsoft created a far better challenge-response mechanism called "NTLMv2." There's also Kerberos, an authentication method used most in the time in Active Directory, and that appeared with Windows 2000 in early 2000. But even a network with the most state-of-the-art stuff will sometimes fall back from Kerberos to either LM, NTLM, or NTLMv2. That's not really a wonderful thing, as Kerberos is considerably more secure than the other three, but there is no way to banish the "LM family" altogether. But if you want to keep your network secure, then it'd be best to ensure that if

Kerberos isn't available then your systems should use NTLMv2 rather than the older, less secure challenge-response mechanisms.

If NTLMv2's been around for nearly 10 years, why aren't we all using it now? Why hasn't LM and NTLM been banished for years? It's the same story as before: compatibility. If you've got some older computers in your network (Windows 9x, for example), then it's some work getting them to talk NTLMv2; MS-DOS systems and Windows for Workgroup systems will never talk NTLMv2. Alternatively, if you've got third-party network-attached devices like some of the Network Attached Storage (NAS) boxes like, for example, an old Quantum Snap Server, then a device like that might not *understand* NTLMv2.

Every modern copy of Windows knows how to log on using either the LM, NTLM, or NTLMv2 challenge-response methods, but which of those methods does it choose? You configure that with a setting in Security Options, "Network security: LAN Manager authentication level." By default an XP box will, when offered a logon challenge, compute *two* responses: the LM and NTLM response. As both of those responses are encrypted with an encryption algorithm that has been cracked in the past and gets easier to crack with every passing year as CPUs get faster, automatically responding LM and NTLM responses becomes less and less ideal all the time. Vista's default is different, and by default it offers the NTLMv2 response.

Will this new default cause you trouble? Well, any 2000, XP, 2003, or Vista system acting as a server can understand and use an NTLMv2 response, so if your Vista clients always produce NTLMv2 responses then you'll probably be okay. But, again, check your network-attached devices, like those convenient little print server things the size of a deck of cards that will let you put a printer anywhere and make it available on the network. As with all hardening processes, testing is important.

Vista also hardens systems a bit by keeping your systems from creating something called "LM hashes." Whenever you type a password into your computer, the computer doesn't store your password anywhere. Instead, it takes you password and subjects it to a mathematical function called a "hashing function" that reduces the password, no matter what length, to a 128-bit number. *That's* what gets stored on your computer and on your domain controllers, not your password; instead, the "password hash" is stored. Why do that? Because if someone gets your hash, then reversing the hash process and figuring out your password just by looking at the hash *should* be nearly impossible, *if the system's designed right*. Microsoft first started doing this back in the LAN Manager days of the late 1980s, and the hashes created and stored by LAN Manager are known as the "LM hashes." In designing the method of storing LM hashes, Microsoft built a fairly good hashing system given the power of computers at the time, but they made one serious error. By looking at the LM hash of someone's password, anyone can instantly determine whether the password that resulted in the hash was fewer than eight characters. Being able to instantly be certain that a particular password was seven characters or fewer is a powerful tool in the hands of bad guys.

With the advent of NT 3.1 in 1993, Microsoft used a different and more secure method of hashing. But Microsoft needed to worry about backward compatibility, and so whenever you changed your password, then NT created and stored *two* hashes: the LM hash and an "NT hash." Remember, if a bad guy gets your hashes, then he doesn't need to crack them both—cracking the LM hash will give him your password without any need of help from the NT hash. Storing LM hashes is potentially security nitroglycerine, but Windows XP and 2003 still

create LM hashes by default. There's been a setting in Security Options, "Network security: Do not store LAN Manager hash value on next password change" in Windows for years, but Microsoft's disabled it by default because, again, older operating systems and network-attached devices may fail if they can't get LM hashes. That changes with Vista.

Personally, I shut LM hashes off my network in 2002 and haven't missed them, and I think that if you can tell your systems to stop creating LM hashes, then you should jump on it—but, again, I would strongly suggest doing some testing first. Vista, however, takes a stronger position and is the first version of Windows to shut off LM hashes right out of the box.

No More Unsigned Driver Warnings

Just about anyone who's ever worked with 2000, XP, or 2003 for any time at all has had to load a driver for a new piece of hardware, or perhaps had to update a driver on an existing piece of hardware. A good portion of the time, trying to install that driver meets with a dialog box containing a dire-sounding message to the effect that you are trying to install a driver that has not been signed by Microsoft's Windows Hardware Quality Labs (WHQL, pronounced "whickel" so as to rhyme with "nickel."). This, the message seems to intone, is a perilously foolish idea but, hey, it's a free country and you are certainly *welcome* to destroy your operating system; *just don't say we didn't warn you*, it seems to say.

No, that's not the exact text of the dialog box, but it's the spirit. You see, Microsoft's WHQL labs offer a set of rules to follow when writing a driver or for that matter anything that contains low-level, "kernel mode" code. If you follow those rules to write the driver, and then buy a digital certificate to sign the driver, then you can submit it to Microsoft's WHQL labs to be tested for compatibility. (As of August 2006, the test costs $250 per driver per operating system; "XP" covers all varieties of XP, so submitting a driver for signing for both XP and 2003 would cost $500, according to "Global WHQL Policies Document," found at `http://www.microsoft.com/whdc/whql/policies/default.mspx`.) If the driver passes, then WHQL signs your driver with Microsoft's certificate. If Windows sees that signature when you try to install a driver, then you don't get the baleful-sounding warning message.

Most hardware vendors don't particularly want to have to pay Microsoft $250 every time the hardware vendor updates a driver, and so they don't bother, kicking off the warning message. I honestly don't know enough about how thorough the WHQL guys are to intelligently comment about whether or not this whole WHQL signing thing is just a revenue stream for Microsoft, or a seriously good idea. In any case, the only reason that you ever see the warning is because of a Security Options setting "Devices: Unsigned driver installation behavior." On 2000, XP, and 2003, it's got three options: "Do not allow installation," "Warn but allow installation," or "Silently succeed." The default was "Warn but allow installation." But if you look for "Devices: Unsigned driver installation behavior" in Security Options on a Vista system, you'll find that it's gone altogether. That's because Microsoft decided to take a hard line on drivers for the 64-bit version of Vista, and require that all 64-bit drivers for Vista be signed by WHQL. That's a pretty stringent requirement in my opinion, and that's not just a bystander's point of view—remember, I'm a 64-bit kinda guy, and that driver thing is in my top five reasons why I might not be able to roll out Vista as quickly as I'd like. In any case, you can read more about this 64-bit driver issue in detail in Chapter 6.

But wait a minute; the Security Options item goes away, and 64-bit Vista requires signing; what about 32-bit? Well, inasmuch as 32-bit Vista is, according to Microsoft, the last 32-bit operating system that they'll ever create, I guess they decided not to worry quite as much about security (which is the supposed reason for driver signing) and just tell 32-bit Vista to allow unsigned driver installs to silently succeed. It's a good change in my opinion, as I have never, ever seen anyone *not* load a driver because of the dialog box's message, and I *have* seen the "are you sure?" dialog get in the way of some unattended installs.

Encryption News

Security can't work without encryption, and of course Microsoft operating systems (except for MS-DOS) have all included some kind of encryption since Microsoft released OS/2 1.0 in 1987. But over the years, the sort of encryption that Microsoft builds into its OSes, and what it does with them, changes. Here are few notes on new crypto capabilities in Vista.

Vista Includes New Cryptographic Services

Every software vendor has to make the choice about whether to try creating its own encryption algorithms or to employ standard algorithms. It might seem at first glance that a software vendor would be better off building their own encryption algorithm and keeping its inner workings secret, but according to security expert Bruce Schneier, writing in his book *Secrets and Lies: Digital Security in a Networked World* (Wiley, 2000), the better route is not to build crypto algorithms that are studied and cross-checked by a handful of insiders, but instead to use a crypto algorithm that's been reviewed by hundreds of mathematical experts. In his book Schneier took Microsoft to task for this, claiming that every single time that Microsoft creates a proprietary cryptographic algorithm, it's cracked in just a few months.

I don't know if that *always* happens, but it's surely happened enough. Maybe that's why Microsoft's using more and more standard cryptographic algorithms. (Maybe they read Schneier's book?) Two that come to mind are the Secure Hashing Algorithm (SHA) and the Advanced Encryption System (AES). Both were developed under the aegis of the U.S. government's National Institute for Standards and Technology (NIST) with the intention of providing a well-thought-out set of algorithms for hashing (SHA) and encryption (AES). AES seems well thought of in the crypto community, but SHA has been attacked successfully in some specialized situations. The most recent version of SHA, "SHA-2," has not been successfully attacked as I write this.

Microsoft has had AES built into XP since SP1 and 2003 since its original release, but only in limited use; as far as I know, the only use XP had for AES was in the Encrypting File System (EFS). With Vista, Microsoft says that you will be able to use AES for encryption with IPsec. Granted, it's not earth-shaking, as previously only offered Triple DES (Data Encryption Standard), and cracking TDES probably won't be practical for some time, but it's a step ahead. Adding SHA-2 to IPsec will also be good, but I should note that as I write this, the group policy

interface does not show options for either AES or SHA-2. I *can* confirm, however, that another Windows technology, BitLocker Full Volume Encryption, does indeed use AES in 128-bit and 256-bit encryption. (You can read more about BitLocker in Chapter 5.)

You Can Encrypt Your Pagefile

Here's good news for the completely paranoid: you can encrypt your pagefile. Just take my advice…don't. Not unless you want to wait, say, an hour or so every time you turn your computer on while you wait for it to decrypt a gigabyte or so of pagefile.

Offline Files Folders Are Encrypted per User

Offline Files is a great technology that allows you to cache data from oft-used file shares locally. It first appeared in Windows 2000 and while it's not for everyone, lots of people like it. But once details of how Offline Files works got out, people soon realized that it presented something of a security hole. You see, in Windows 2000, all of the cached files were stored in a directory easily viewed by any user. Thus, if I shared a computer with you and you used Offline Files, then I could poke around the folder holding the cached files—everyone on the same machine shared the same folder—and that might not be good.

When XP came around, Microsoft encrypted the folder that held the cached Offline Files data. But the process that did the encrypting was a service that ran as the LocalSystem account, which meant that the EFS encryption key for the Offline Files data was easily utilized by anyone running as LocalSystem. Unfortunately, it turned out to be really easy to log on as LocalSystem—just use the `at.exe` scheduler program to start up a command prompt; as the scheduler program runs as LocalSystem, you get a command prompt running under the LocalSystem account—cracking Offline Files to peek into the cached files of someone who shares your machine was still relatively easy.

Vista changes that in two ways. First of all, everyone's cached files are cached with *their* EFS key, not LocalSystem's. Second, even if Microsoft *hadn't* changed that about the operating system, it'd still be pretty tough to exploit, as logging on as LocalSystem has gotten a lot harder. All of the old tricks that I've been able to use in the past to log on as LocalSystem no longer work in Vista!

New Event Viewer

Here's a good surprise: the Event Viewer has had a complete reengineering. The new Event Viewer:

- Can collect events from many systems to one system's log, allowing you to centralize event logs.
- Lets you easily tell it what to do if particular events occur, like telling it to send you an e-mail, run a program, reboot a system, or the like.

- Allows you to create custom queries so you can essentially tweak Event Viewer to show you just the things that you want to see.
- Event Viewer Reports its data in XML.

It's beyond the scope of this book to go into Event Viewer in detail, but I'd like to show you a bit of what I think you'll like about the new Viewer, in ascending order of importance from my point of view. But before we go any further, let me show you how to start Event Viewer. As with all Windows things, there are several ways.

- If you reenabled the Start ➢ Run… command as I suggested earlier in this chapter, just click Start/Run… and then fill in **`eventvwr`** and click OK.
- If you restored Administrative Tools to your Start menu, then just click Start ➢ Administrative Tools ➢ Event Viewer.
- Alternatively you'll need to do a little spelunking in Control Panel: click Start ➢ Control Panel ➢ System and Maintenance and, under "Administrative Tools," click "View event logs."

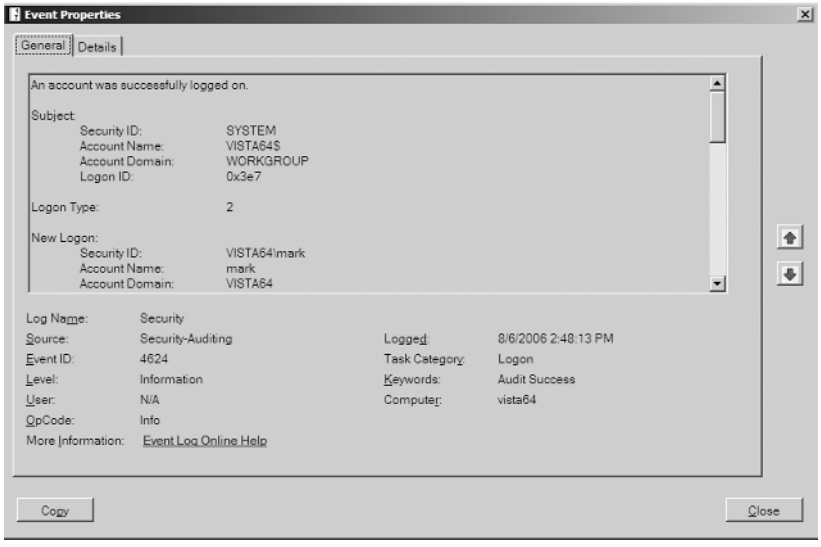
As some have observed, "Vista: it's everyone's 10 favorite user interfaces."

XML Format Comes to Event Viewer

Woo, hoo, XML! Wait, wait, don't flip the page…

Yes, I know, you've heard the abbreviation "XML" far too often, but here's a case where you'll like it. Let's take an example event, a simple security event that reports that the system's time was successfully changed. The event looks graphically like Figure 1.9.

FIGURE 1.9 Typical Vista event



This is an event generated in the Security log because I've just logged on with a username of "Mark." You'll note at first that Event Viewer presents the event in a different format than the one that we've seen since NT 3.1. Notice that there's a button that's actually labeled "Copy" instead of hoping that you just somehow know that the button on the XP Event Viewer that looks like two pieces of paper means "click this and the relevant stuff from this event will be copied in ASCII text format to the Clipboard."

You can't see it because the text in the box is pretty big and scrolled out of the visual part of the box, but here's some of it:

```
This event is generated when a logon session is created. It is generated on the
computer that was accessed.

The subject fields indicate the account on the local system which requested the
logon. This is most commonly a service such as the Server service, or a local
process such as Winlogon.exe or Services.exe.

The logon type field indicates the kind of logon that occurred. The most common
types are 2 (interactive) and 3 (network).

The New Logon fields indicate the account for whom the new logon was created,
i.e. the account that was logged on...
```

I've looked at "logon type" in logon events in Windows for years and never bothered to look up what a "logon type" is. In contrast, Vista tells me. I didn't copy all of the information from the message, but it goes on to explain what a logon GUID, package name, and bunch of otherwise-cryptic stuff was. Very nice. In another event, one reporting that the system time changed, the explanatory text goes on to basically say "look, system time updates happen normally, so you can probably ignore it. But there are reasons why bad guys might try to change a system time."

If you click the Copy button and paste the results into Notepad, you get a lot of information. It starts off looking like event logs have always looked, telling you where the event came from, its ID and so on. Then it starts with a bunch of XML, which, shortened a bit, looks like

```
<Data Name="TargetUserSid">S-1-5-21 …-1000</Data>
<Data Name="TargetUserName">mark</Data>
<Data Name="TargetDomainName">VISTA64</Data>
<Data Name="TargetLogonId">0x20788</Data>
<Data Name="LogonType">2</Data>
<Data Name="LogonProcessName">User32 </Data>
```

What's *this* mess? It's all of the pieces of data specific to this particular event log entry. Now, event log items have always kept this kind of information, but it hasn't been very useful for two reasons. First, it's been hard to export this kind of data out of the event logs for use in scripts or other homegrown tools that might increase the value of the data in the event logs, and, second, it's hard to know what the data means. You can fairly easily write a script that

grabs "data item 1," "data item 2," "data item 3," and so on from a given event, but what *is* data item 1? Well, it depends on what particular event you're looking at. But not with Vista. By storing the data in XML and making that data easy to get to from scripts or even right from the Event Viewer—you can right-click any log in Vista and choose "Save events as," and you're offered to save the files as either text files, XML files, or comma separated variable files—then not only can you get to the data, but the data identifies itself. That's the beauty of XML; the line

```
<Data Name="TargetUserName">mark</Data>
```

is XML-ese for "there's a piece of data in this event named the 'TargetUserName,' and the value of that data in this case is 'mark.'" But you needn't export the data to look at it; you can always click the Details view in any given event log entry to see something like Figure 1.10.

A moment's glance will show you, I guarantee you, that this will be a jumping-off point for a lot of neat event log management tools. But this was the least interesting of the three things that I wanted to cover about Event Viewer; let's move on to the next one.

Custom Queries Lets You Customize Event Viewer

It's always been possible to filter items in Event Viewer in a simple way by right-clicking in the Event Log, choosing New Log View, and then adjusting its filter properties. But Vista's Event Viewer takes it a bit further. To see how, take a look at the Event Viewer when started up in Figure 1.11.

FIGURE 1.10 An event log entry's details

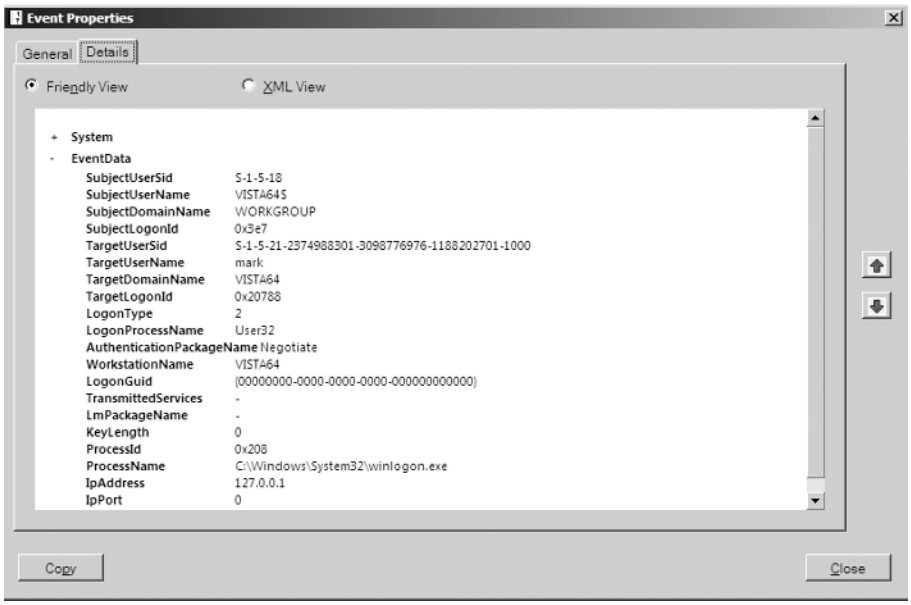
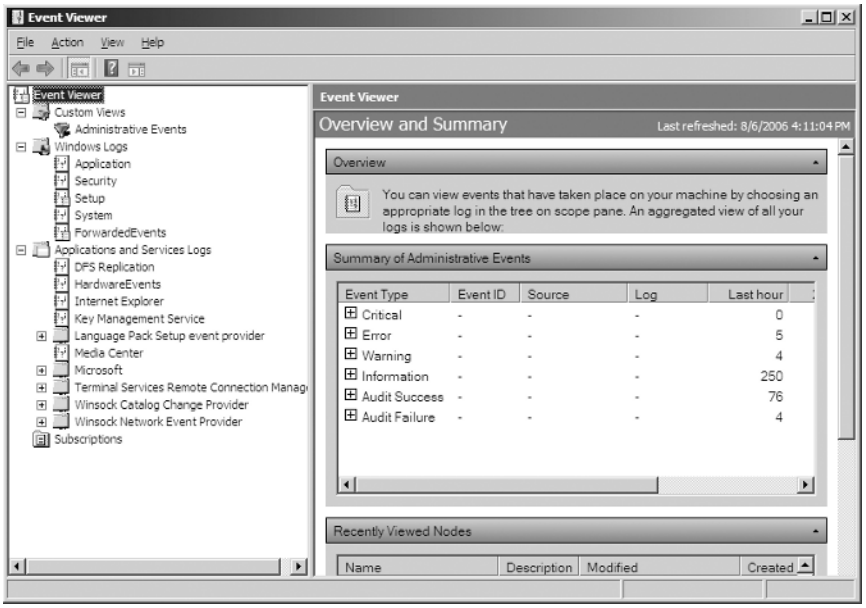


FIGURE 1.11 The Event Viewer



Like the old Event Viewer, you get a pane down the left-hand side listing the logs that you can peruse. But instead of the standard Application, System and Security, Vista's Event Viewer fine-tunes your events into dozens of smaller "sub-logs." You can see in its right-hand pane a summary of entries and, you'll note, there are more levels of event than Information, Warning, Error, Audit Success, and Audit Failure; now there's also Critical. But look in the upper left-hand corner and you'll notice a folder called "Custom Views" and, inside that, a folder named "Administrative Events."

I didn't create that, it was already built in Vista. It collects all of the events from all logs that are Critical, Error, or Warning. In short, it's one-stop-shopping for keeping an eye on what's broken. But what if we wanted the "auugh! log," a collection of just the Critical stuff? Simplicity itself. Just right-click the Custom Views folder and choose "Create custom view…" and you'll see a dialog like the one in Figure 1.12.

With this dialog, it's simple to see how Microsoft prebuilt the "Administrative Events" log. To create a "Criticals only" log, I'd change the dialog like so:

- Leave "Logged:" as "Any time;" this means to show any events in the log. (Remember that by default Windows only keeps as many events as it has storage to hold.)
- In "Event level:," check only "Critical."
- Choose the radio button between "Event log:" and "By log," and click the drop-down box to the right of them. Check the boxes next to "Windows Logs" and "Applications and Services Logs" to choose all logs.
- Click OK and when you get the dialog that says that this might be a bit slow and are you sure, click "Yes." You'll see a dialog like Figure 1.13.

FIGURE 1.12 Creating a custom view in Vista's Event Viewer

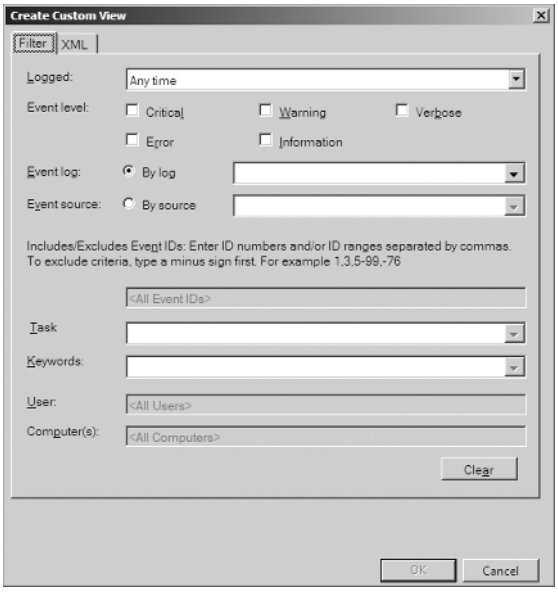
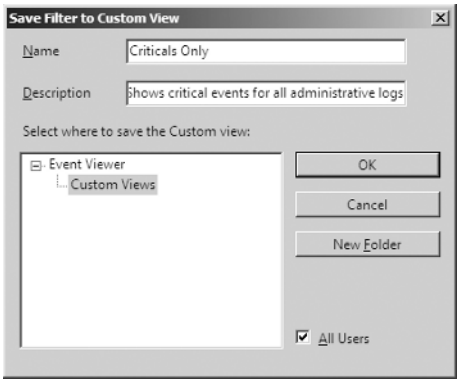


FIGURE 1.13 Name your new custom view



Here, I've filled in "Criticals Only" in the Name: field, and "Shows critical events for all administrative logs" in the Description: field. Click OK and you'll see the new view. And by the way, that's not a silly example. After running just a few days, my Vista system has generated tons of event log entries of varying levels of importance. But the "Criticals Only" log has just a dozen events in it, and they were all interesting. (My favorite was a message telling me that a particular program was "slowing down the Windows Shell," presumably meaning that shutting off this badly written program, whatever its name was, would make things faster. The program? `Explorer.exe`. Who says programmers lack senses of humor?)

Once you've created your ideal custom view, it's easy to back it up or spread it around. Just right-click it and export it. And guess what kind of file it creates? Yup, you got it: XML. (Perhaps they should have named Vista "Windows XML?" Then it would have sounded more like an upgrade from "XP.")

Generating Actions from Events

XP and 2003 brought a really nice feature called "event triggers." The idea was that you could use a command-line tool called "`eventtriggers.exe`" to instruct the Event Log service that if a particular kind of event occurred then the Event Log service would start the application of your choosing. Not many people seemed to discover it, but I wrote about it in a few magazine articles and suggested that you could build a pretty neat system for alerting you to problems in the network. There were three ingredients:

- You'd need a cell phone that could receive text messages via e-mail. For example, my cell carrier is Verizon Wireless, and you can send an SMS text message to any Verizon cell phone by sending e-mail to *cellphonenumber*@vtext.com.
- You need a program that can send simple e-mails from the command line. There's a free one called "blat" at `http://www.blat.org`.
- You need XP or 2003, as they support event triggers.

I put this all together by suggesting that if there were particular events that you were concerned about—say, an account lockout happened—then you could use `eventtriggers.exe` to tell the Event Log service, "if an account lockout happens, run such-and-such blat command line to send me an alert on my phone as a text message. It worked pretty nicely but was, admittedly, cumbersome. So the new "Attach task to event…" option is a real blessing.

To see this in action, open up the Application log and look at the events in it. If this is your first look into Vista's Event Viewer, look in the folder "Windows Logs"—it's probably already open, if not then open it—and notice that these logs bear the familiar names of Application, Security, and System, as well as two new ones named "Setup" and "ForwardedEvents." Click the Application folder in the left-hand pane and in the right-hand pane (I always close the Action pane because I think you'd need a computer with a screen that isn't just in "landscape" mode, you'd need one in "panoramic mode" in order to make use of MMC 3.0's three panes) you'll see the events in that log.

Right-click any one of them and you'll see in the resulting context menu that you've got a new option, "Attach Task To This Event…;" click that, and you'll see a wizard page like the one in Figure 1.14.

Why a wizard? Well, as it turns out, Vista's Event Viewer offers you several options on how to respond. (They even simplified setting up my suggestion about e-mailing admins when an event occurs, as you'll see.) Click Next to see a figure like Figure 1.15.

First, as with `eventtriggers.exe`, you can specify any given application. Or you can send an e-mail, or display a message on the server's desktop. I'll consider all three options in a moment, but for now, I'll click the radio button next to "Send an e-mail" and then Next to see something like Figure 1.16.

FIGURE 1.14 Starting the Create Basic Task Wizard

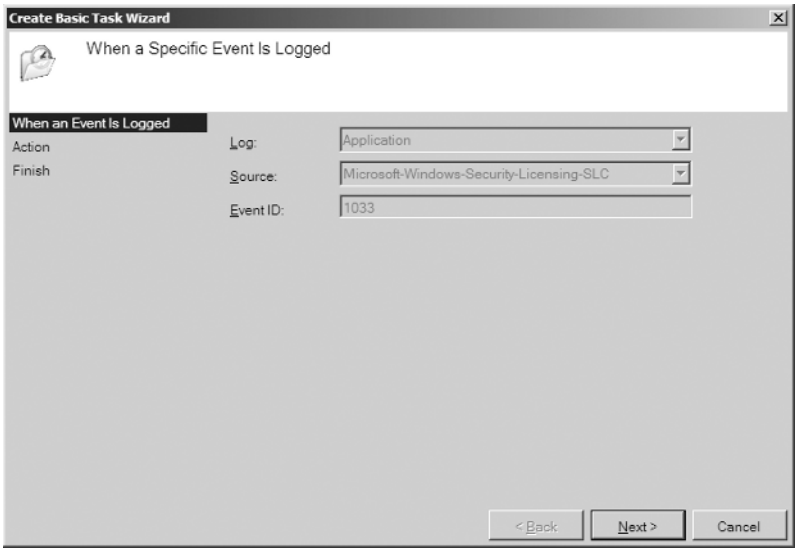


FIGURE 1.15 Event Viewer offers three kinds of responses.

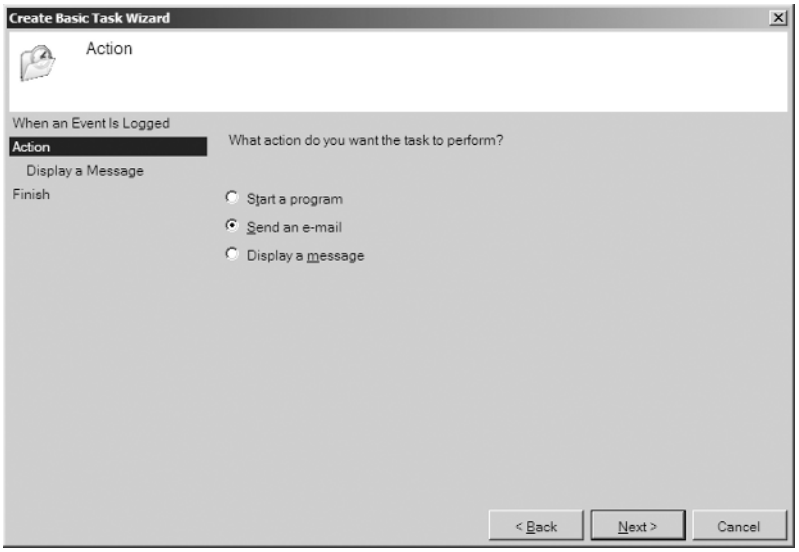
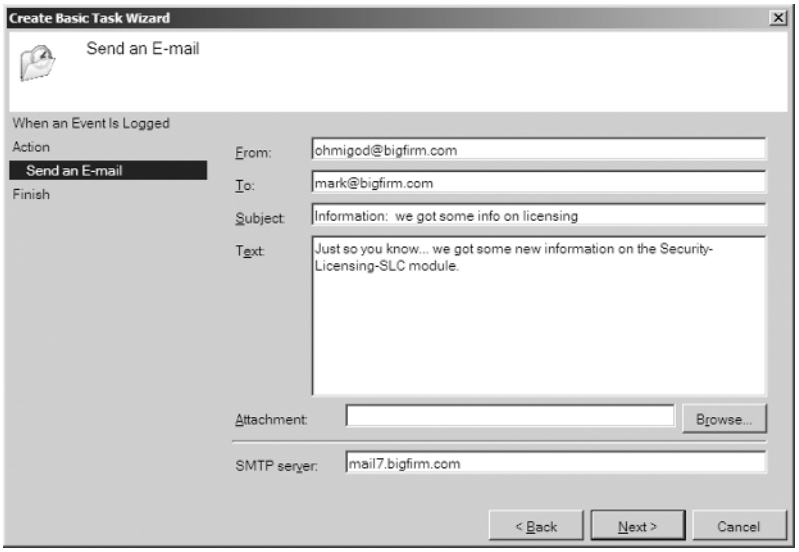


FIGURE 1.16 Setting up an e-mail notification



This page looks very much as you'd expect, allowing you to punch in a from address, to address, subject, and text. It even lets you add an attachment, which is a nice touch, and specify the name of the SMTP server to use to send the e-mail.

Be sure to configure the SMTP server to accept e-mails from this server, or you'll never get an alert via e-mail. All well-configured SMTP servers nowadays have strict rules restricting SMTP relaying and would probably reject the e-mail that the Event Log service tried to send to the SMTP server. And setting up random extra SMTP servers without all of those strict rules is a *really* bad idea, as it's one way that spammers send all of that junk but don't get caught.

If I click Next, I get a summary screen like the one in Figure 1.17.

This is a nice summary of what's going to happen once I click Finish, although truthfully it's not necessary. An administrator can always modify or delete an event task, as you'd expect. Ah, but *where* you modify or delete that event task, that'll surprise you. When I click Finish, I get the message box in Figure 1.18.

FIGURE 1.17 Summarizing the trigger

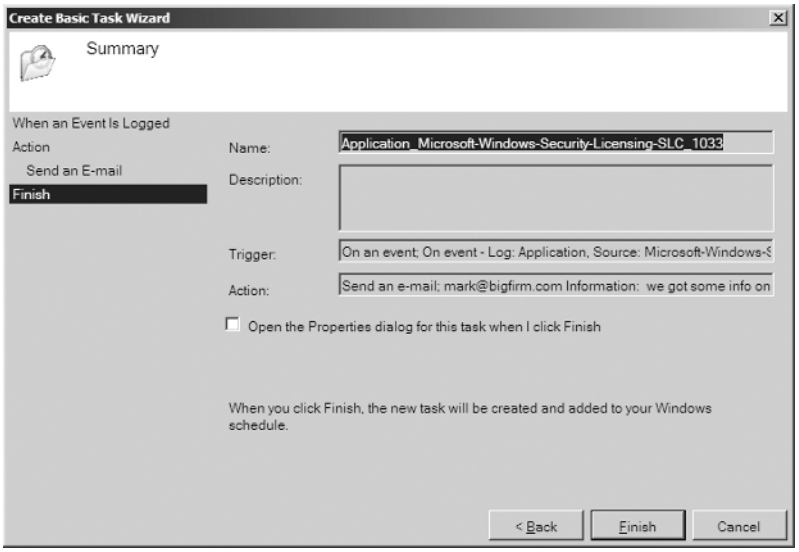
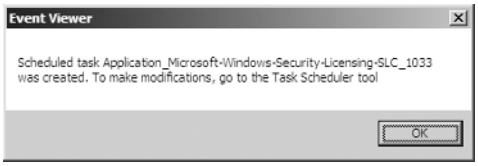


FIGURE 1.18 Changes? Off to the Task Scheduler



This seems like a bad idea to me. Vista's user interface does a fairly decent job of providing what Microsoft has come to like calling "discoverability," which is their recently coined term for "a user interface that makes figuring out what you can do with a GUI program easier." So here you've created an event task in the Event Viewer; you'd think that you could modify or delete it in the Event Viewer. But no, instead Microsoft's got you going to the Task Scheduler to do that.

Telling the Event Log Service to Display Messages

I highlighted sending a piece of mail, but Vista's Event Viewer offers a couple of other options as well, as you may recall from Figure 1.17. The "Start a program" option is pretty straightforward, so I needn't really discuss that, given that I'm not covering this in minute detail, but I wanted to talk about the last option: "Display a message." On its face, it's pretty simple; it just asks you what text to display. When the corresponding event occurs, the Event Log service

just pops up a message box with whatever text you specified. But the fact that the option existed puzzled me, at first. What real good did this offer? But then it hit me.

You see, ever since I started with NT, I've always had a bit of trouble getting used to the "chatter level" that NT offers. I'd spent years working with various microcomputer operating systems designed with the point of view that well, heck, if they're having a problem, no matter how small, then they want *you* to know about it, and *now* with at minimum a message box and at maximum a "system modal dialog," one of those big annoying things that can't be resized or minimized. And, in fact, some of Windows' behavior is still like that, but in an uneven fashion. If a document doesn't print, then you get a big can't-miss-it pop-up in the system tray to that effect. Windows applications are just as bad as Windows itself, and I offer Adobe Acrobat Reader as Exhibit A. When Adobe Reader decides that you're gonna get an update, then while you're getting it, Reader puts this huge dialog box on your screen with status information, a dialog box without a "minimize" button.

But what's always puzzled me about Windows' "chatter level" is that there are things that it *doesn't* say a blessed thing about, but that might be useful. For example, I had a workstation a few years ago whose hard drive failed. I got it running long enough to get what I needed off of it and, as I waited for the files to copy, I noticed that the Event Viewer had been logging disk failures for weeks. "You'd think *that* might merit a pop-up box," I grumbled to myself. There are, similarly, a handful of Active Directory events that are downright scary, things that portend long-term unhappiness if not seen to posthaste, but does AD pop up a dire warning message? No, instead AD just quietly issues an event, sort of like an unhappy employee petulantly writing memos "to the record" to cover his backside when his company turns out to be the next Enron—"I told them so," he anticipates saying with disguised glee. (Okay, in reality I imagine that the AD coders assume—probably correctly—that, again, most domain controllers don't have people sitting and watching the DC's desktops all the day long.)

Okay, so why *is* Windows so schizophrenic about what it shouts and what it whispers? Simple: it's not like there's a central User Interface Message Approval Board at Microsoft. If Janet's working on the file sharing client and Peter's working on the shell hardware detection module (the thing that pops up a box asking you what to do every time you plug a USB device into your computer), then Janet gets to control the degree of shrillness or demureness the file sharing client displays to the user, and Peter likewise can dial up or down the hysteria level of the warnings and errors that the shell hardware detection module displays. (In my opinion Peter needs to consider more decaf, as the fireworks that start every time a USB stick drive is plugged into a system is a bit overmuch.)

The cool thing about the "Display a message" option is that it lets you increase Windows' hysteria level about any event or events that you choose. Never again will I miss a disk error that formerly would have secreted itself in the logs and, when Server 2007 comes out with the same Event Viewer, never again will I pass over a message to the effect that a particular domain controller hasn't talked to the rest of the family for a few weeks.

Forwarding Events from One Computer to Another

I imagine that I needn't explain my reasoning in saying that my number one favorite new Event Viewer feature will probably be everyone else's favorite new Event Viewer feature: event forwarding. It's always been true that NT systems have had event logs, and that those event logs are completely decentralized: if you've got 500 XP boxes that someone's been trying to hack and you want to look at all of the failed logon attempts for all of those 500 boxes…what do you do?

Well, you could walk around to 500 systems, or buy a third-party tool, or use a serviceable and nicely priced—free—but slightly rough-edged tool called eventcombmt. It's sort of amazing, but after 13 years of NT, we still don't have a built-in method of centralizing events from the event logs of scattered machines. Unless, of course, you've got Vista. It offers something called "event forwarding" that lets you collect events from other Vista systems to one single system. Just tell all of those 500 systems to report logon failure events to one of their own, and then all you need do is to sit down at that system, open up a folder called "ForwardedEvents," and it's all in one place.

So is it just Vista talking to Vista? I attended a briefing where a program manager for Vista's Event Viewer said that he thought it very likely that they'd have modules that you could add to earlier versions of Windows (I'm guessing XP and 2003) that would make it possible for a Vista system to collect events from those older OSes, so the answer seems to be "right now yes, it's just Vista, but there's a chance that earlier Windows may get event forwarders." But don't hold your breath. Back in 2002, Microsoft promised a free tool that would collect all of the Security events from 2000, XP, and 2003 systems into a centralized SQL Server. When they finally finished in late 2005, they said "oh, we changed our minds," and stuck it into Microsoft Operations Manager, which you probably know is most definitely *not* free.

Subscription Overview

Another Microsoft phrase for event forwarding is "subscription"—one computer "subscribes" to events from other computers. Here's basically how it works, in Microsoft "subscription terminology."

- In a *subscription*, you designate one Vista system to collect events from one or more other computers.
- That system is called *the collector*.
- The systems from which the collector gets events are called the *source* computers.

Once you set up a subscription, the collector then *polls* the sources every 15 minutes or so for new events. By default, those events go into a folder called "ForwardedEvents" but that can be changed. And one more thing: subscriptions work a lot easier if the collector and the sources are in the same domain. It's possible to make it work between two systems in a workgroup, but it's more trouble. How you make all of this work varies, but here's the overview of how to set up a subscription in the easiest way.

Set Up the Sources

First, go to every computer that will be a source computer and enable a service that listens for the collector's requests for events. You do that with a command-line tool named `winrm`. Also, adjust the permissions on the event logs of all of the sources so that when the collector asks for those logs, then the sources are willing to cough up events.

Set Up the Collector

Then, enable the software on the collector so that it can, well, collect. Every Vista box comes with it, and it's called the "Windows Event Collector" service. Just start it and tell it to start automatically henceforth, as you'd do with any service. (Actually you needn't even do that because, as you'll see, the first time that you try to create a subscription, the Event Viewer asks you if you'd like it to configure the Windows Event Collector service for you.) Then create a new subscription. In that subscription, you tell the collector exactly which events to collect, and the names of the systems to request them from. In a little while, the events will start flowing into the collector.

Creating an Example Subscription

Let's walk through setting up a simple subscription. In this example, I'll have the following settings. If you'd like to follow along, just duplicate this setup:

- Two Vista systems; one's named "Vista1" and the other will be "Vista2."
- Both Vista1 and Vista2 are members of an Active Directory domain named "`bigfirm.com`." I'm doing that because it makes things a bit easier than connecting two systems in a workgroup would be.
- When configuring both systems, I'll be logging on to them as an account named "mark@bigfirm.com," which is a member of the Domain Admins group.

My goal for this simple demonstration is to have the Vista1 machine act as the collector and the Vista2 machine act as its sole source. I'll have Vista1 collect all of the information-level Windows events from Vista2 machine. As I outlined before, I'll do this in two steps: first I'll visit Vista2, the source, to enable some services on it and adjust its security so that it'll let Vista1 extract events from it. Then I'll move over to Vista1, the collector, to turn on the Windows Event Collection service and create the subscription.

When we start out, I've got the two Vista systems running basically in an out-of-the-box configuration, although I've given them a very simple display theme to accommodate screen shots that are easier to read. Both have Windows Firewall up and, as is usual for Vista, WF by default has no exceptions enabled except for what it calls "core networking."

Step One: Set Up Vista2 for WinRM

First, we'll log onto Vista2, the system sending the events to the collector, and fire up the infrastructure necessary to allow it to respond to subscription requests from Vista1. In a sense, the fact that Vista2 is responding to requests from Vista1 for events makes Vista2, the source, something of a server—and so we'll have to start up a service to respond to those needs.

Microsoft built the event log stuff on top of a set of remote desktop management standards called "WS-Management," and Microsoft's name for their implementation of it is "WinRM," for "Windows Remote Management." Just fire up an elevated command prompt (right-click Command Prompt and choose Run as administrator, as always, and go ahead and respond as necessary to the UAC prompts), and type

```
winrm quickconfig
```

The Windows Remote Management Tool will respond like so:

```
WinRM is not set up to allow remote access to this machine for management.
The following changes must be made:

Set the WinRM service type to delayed auto start.
Start the WinRM service.
Create a WinRM listener on HTTP://* to accept WS-Man requests to any IP on this
machine.
Enable the WinRM firewall exception.

Make these changes [y/n]?
```

Yes, you read that right: WS-Management runs on port 80 (that's the reference to "HTTP://*") but no, you're not setting up a web server on your Vista box, at least not a full-scale one. In response, I type y and Enter to get this response:

```
WinRM has been updated for remote management.
WinRM service type changed successfully.
WinRM service started.
Created a WinRM listener on HTTP://* to accept WS-Man requests to any IP on
this machine.
WinRM firewall exception enabled.
```

Once that's done, stay at the command prompt and give the collector computer the permission to look at events on this computer. You do that by putting the collector computer's computer account into each of the source computers' "Event Log Reader" groups—and as you've probably guessed, the "Event Log Readers" group is new to Vista. We could do this with the GUI, but quite honestly I find it easier to do from the command line. There's a NET (not a .NET) command that's been around NT for over a decade that lets you put a user account into a local group that we'll find useful here. Its syntax looks like

```
net localgroup groupname accountname /add
```

The group's name is, again, "Event Log Readers." But what's the name of the Vista1 computer? Well, we want to enter the name of the Vista1 machine's domain account. Ever since Windows 2000, machines have gotten Active Directory names looking like *machinename$@DNSdomainname*. Thus, a machine named "Vista1" in a domain named

"bigfirm.com" would have a domain account name of "vista1$@bigfirm.com." (The dollar sign suffix fixed a problem that popped up back in the NT 4.0 days whenever a user had the same name as her computer.) The command I'd type at the command line, then, would be

```
net localgroup "Event Log Readers" vista1@bigfirm.com /add
```

To which Vista2 should respond `The command completed successfully`. Vista2's ready, let's move over to Vista1 for Act Two.

Step Two: Create the Subscription on Vista1

Now I move to Vista1 and, again, log on as mark@bigfirm.com, a domain admin. Before we get into configuring Vista1, let's take a moment and verify that we've got connectivity to Vista2, and that it's running the WS-Management software. We can do that with a command-line tool, so start up an elevated command prompt on Vista1 and type

```
winrm id -remote:vista2.bigfirm.com
```

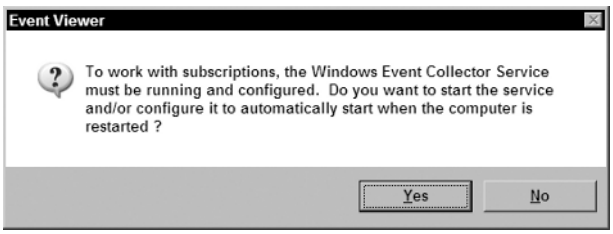
That's the `winrm` command again, but this time we're using the `id` parameter, which is basically nothing more than a "WS-Management ping." The `-remote` option identifies what system to talk to. Vista2 responds in a successful fashion with output that looks something like

```
IdentifyResponse
    ProtocolVersion = http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd
    ProductVendor = Microsoft Corporation
    ProductVersion = OS: 6.0.5472 SP: 0.5 Stack: 1.0
```

In contrast, bad responses would be something like "WS-Management could not connect to the specified destination." So now we know that we've got connectivity; excellent. We're ready, then, to set up Vista1 to collect.

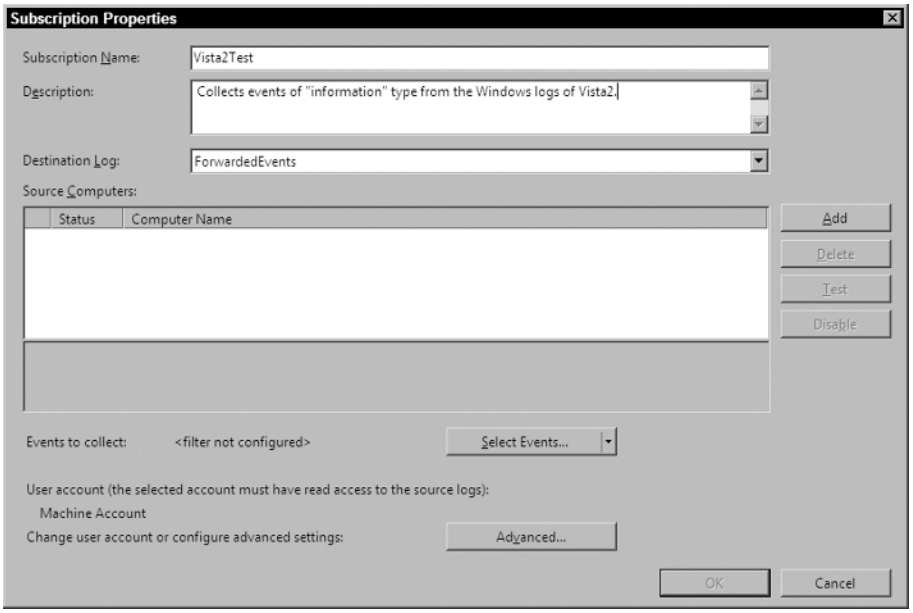
On Vista1, we next start up the Event Viewer. Then we right-click the Subscriptions folder and choose "Create Subscription." As this is the first time that we've told Vista1 to create a subscription, we get a dialog box like the one in Figure 1.19.

FIGURE 1.19 Shall we start the Windows Event Collector service?



Click yes. If you'd wanted to set up the Windows Event Collector service beforehand, then you could have alternatively just opened up an elevated command prompt and typed **`wecutil qc`**. Had we done that, we'd not have gotten the dialog box in Figure 1.19. Anyway, after clicking Yes at the dialog box, we then get the quite formidable-looking Subscription Properties dialog box, as you see in Figure 1.20.

FIGURE 1.20 Creating a subscription, part 1



As you can see, I've started to create the subscription by filling in the "Subscription Name" and "Description" fields.

And do yourself a favor: don't put any spaces in your subscription names. If you do, then some syntax gets ugly later.

But before this subscription will start working, we've got to tell the Event Viewer what kind of events to grab from Vista2 and oh, that's right, we need to tell it to specify that Vista2's the machine to get those events from. First, I'll specify what to look for by clicking the Select Events… button. That'll raise a dialog box like the one in Figure 1.21.

In this Query Filter dialog box, you see that you can choose the "fright level" of the events to grab (Critical, Error, etc.) with the check boxes next to "Error level:." I'll check "Information," as that's what I set out to collect. Above that, you can choose to collect every qualifying event in the source, or you can choose to only get events generated a certain number of days or hours in the past. For now, I'll leave it at "Any time."

Perhaps the biggest choice is "which logs to pull from?" A little browsing in Vista's Event Viewer will show that, as I've already mentioned, Vista has a whole lot more logs than the three or so normally found on a pre-Vista desktop operating system. If I click the drop-down box next to the "Event log:" radio button, I'll see something like Figure 1.22.

FIGURE 1.21 More creation: pick the events to track

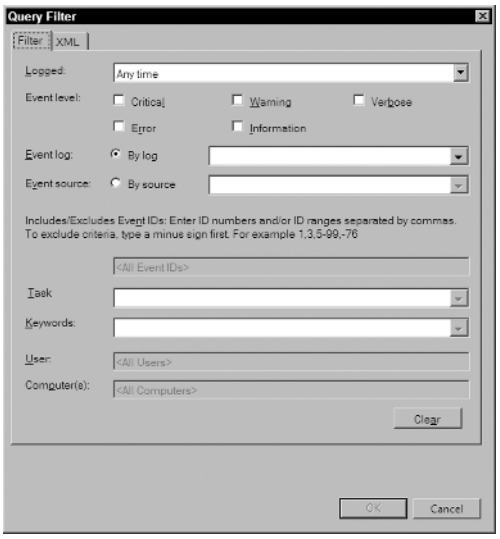
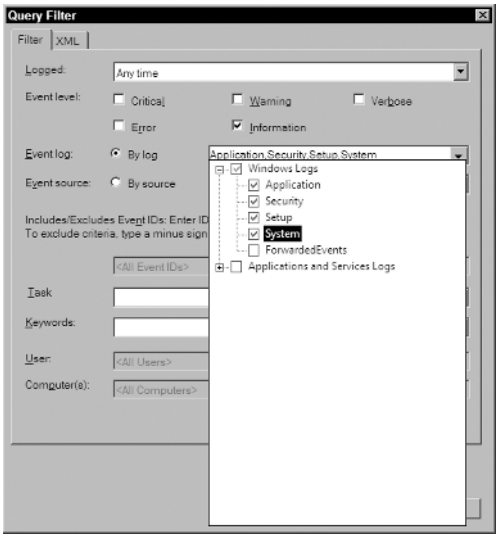


FIGURE 1.22 Just the Windows logs, please

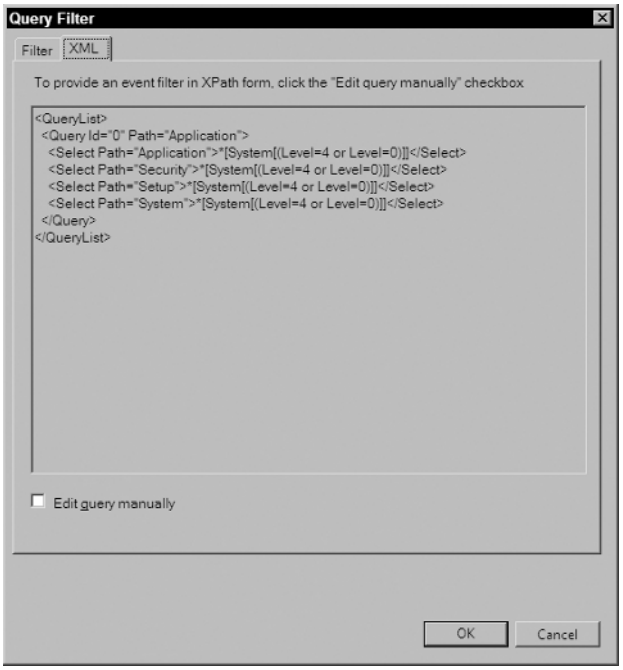


If you're following along, then you'll just see the "Windows Logs" and "Applications and Services Logs;" I deliberately expanded the list a couple of levels to show you what's available. Nothing's checked by default—I checked all of the Windows Logs items except for "ForwardedEvents," as I had nightmare visions of accidentally setting up an infinite loop between Vista1 and Vista2 if I ever happen to set up Vista2 to grab from Vista1, instead of the vice versa situation we're setting up here. If this drop-down looks a little strange, it is—I don't recall seeing a structure like this on a Windows GUI before.

Look more closely back at Figure 1.23 and you'll see that you can *continue* to fine-tune the events that you want to grab. In sum, then, you could, if you wanted, only forward event ID 1030s but only when generated by the Group Policy service. Oh, and remember that XML centrism that we saw in Event Viewer before? You might have noticed the "XML" tab on the Query Filter page. If I click that, I see something like Figure 1.23.

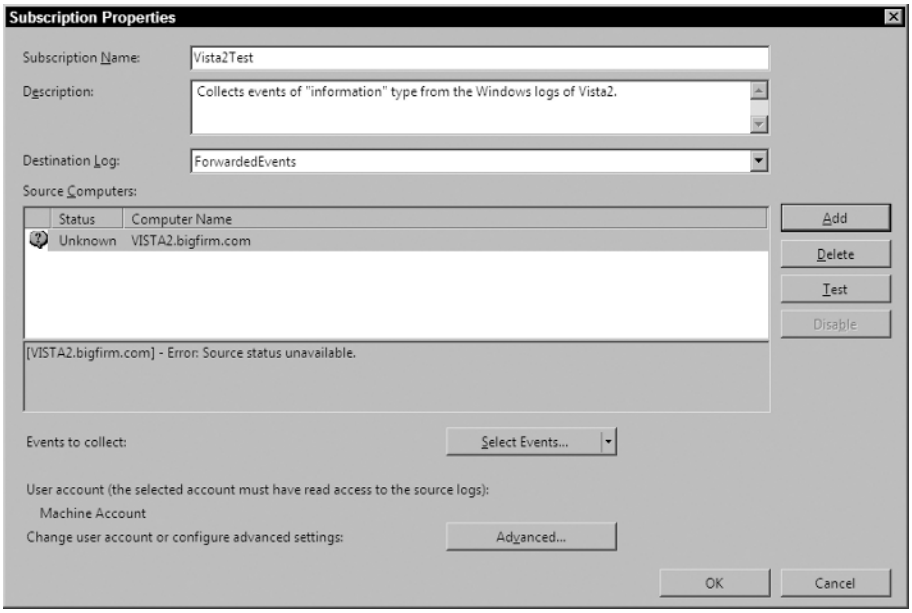
That's something called "XPath," an XML-renderable way of describing a query. Basically what Microsoft's done here is to offer us a moderately easy-to-use GUI to tell the collector what to grab from the source, but, if we really want to cook up an exotic filter, then we can skip the GUI and do it in XML. Or you can spend 30 minutes fiddling around with it in the GUI, but then you'd like to be able to reproduce it on demand…so you just copy the XML to Notepad, save it somewhere, and then you can copy from the Notepad file sometime in the future and paste into the XML tab of some query that you'll create in the future. I'm not getting that fine-grained for this event forwarding tryout, however, so I'll just click OK to clear the Query Filter dialog and return to the Subscription Properties dialog.

FIGURE 1.23 XML-ese for the query



In the Subscription Properties dialog, there's just one more thing I must do before I can take this subscription out for a spin: I've got to tell the Event Viewer where to draw the events from. That's easy; I just click Add and I'll get the standard Select Computers dialog that looks exactly like the one you'd see on an XP box, so I'll skip the screen shot. I just fill in the name "Vista2," click Check Names and then OK to end up with a Subscription Properties dialog looking like Figure 1.24.

FIGURE 1.24 Subscription setup before activation



I included this screen shot because the first time you've finished crafting a subscription, then you'll look at your dialog box one last time before clicking OK…and feel your heart drop. Notice that in the "Source Computers:" field, Vista2's there, with "Unknown" next to it, and, below that, the grim report that says "[VISTA2.bigfirm.com] - Error: Source status unavailable." I mention that to let you know that you really needn't worry if you see that; it appears to be normal. Now that the subscription has been defined, I click OK.

Troubleshooting Subscription Delays

With that done, it's time to view the fruits of our labors. So we open up the ForwardedEvents log in Event Viewer, to see…nothing.

But don't panic; that's apparently normal. Just give it about 15 minutes—Microsoft's golden interval, it seems—and the events start popping up. In fact, that's an important point about subscriptions. Working with them is similar to flying somewhere on an airline in that both have the same motto: "expect delays."

Adjusting the 15 Minutes

Now, you can change some of those delays, but not all. There are two main reasons for the delays that you'll see in Event Log subscriptions. The first reason explains our current wait: "the 15 minute rule." By default, Vista systems acting as event collectors only poll their source systems every 15 minutes. Now, you'd *think* that they'd do the *first* poll the very moment that you clicked that last "OK" to create the subscription, but Vista doesn't; it seems to just wait about 15 minutes before even *starting* to grab that data.

Now, if you like, you can change that 15 minutes to something shorter. You can't do it in the GUI; you need a command-line tool called `wecutil`—Windows Event Collector utility—to do the fiddling needed on the collector. You'll need to run `wecutil` twice. First, we'll use wecutil to tell the Event Log that we don't want the standard 15-minute cycle, and instead we want to customize the subscription. That command looks like `wecutil ss` *`subsname`* `/cm:custom`, where *`subsname`* is the subscription's name. The `/cm` is short for "configuration mode," hence the "custom." If you wanted to return your subscription to its original standard format, you'd just replace the `/cm:custom` with `/cm:normal`. The `ss` stands for "subscription settings." To shift the Vista2Test subscription to "custom" configuration mode, then, we could type

```
wecutil ss vista2test /cm:custom
```

Press Enter, and if you get no response, that's good. Clearly `wecutil` is of the class of command-line tools who are the strong, silent types. Now that we've freed Vista2Test from the iron constraints of "normal," we can change the polling rate. We do that with the command `wecutil ss` *`subsname`* `/hi:`*`milliseconds`*, where *subsname* is, again, the subscription name, and *milliseconds* is how much time to wait between polls, in milliseconds. Try it out by setting it to 10 seconds with this command:

```
wecutil ss vista2test /hi:10000
```

If you're following along and want to try this out, then clearly you'll need a way to create an "information" event on Vista2. The easiest way I've come across is to open a command prompt of elevated privilege and type `ipconfig /renew` to renew Vista2's DHCP lease. That generates an "information" level event on Vista systems. Those events will appear on Vista1's ForwardedEvents folder nearly immediately, although you'll have to press F5 in Event Viewer on Vista1 to see them.

Understanding the "Reboot Delay"

The second reason you'll sometimes see subscriptions change slowly has to do with two services: the `winrm` service that provides the WS-Management capabilities on the side of the source computers, and `wecsvc`, the Windows Event Collector service on the side of the collector computer. Here's what you're likely to see: you boot up one of your source computers,

and go over to the collector to start seeing events from that computer. So you look. And look. You check the parameters of the subscription, which you can do on the collector by typing

```
wecutil gs vista2test
```

If you do that, then you'll see, plain as day, that you've set the polling interval—wecutil calls it the "heartbeat interval"—to 10 seconds. By the time you do that, then clearly 10 seconds will have elapsed. More than a few times. And yet…no events. What's going on?

It's all due to a change that Microsoft made to the way that some services start up. Microsoft is very sensitive to the fact that their operating systems tend to, um, take a long time to boot up. So for the past few years (XP was part of this) they've been trying to find things that they could kind of *remove* from the boot process—technically, anyway—and so make their OSes (particularly the desktop OSes) seem more sprightly. Their latest plan to make Windows seem to boot more quickly involves a new class of services.

If you've been following Windows (well, the NT side of Windows) for a while, then you know that Windows services have different startup classes. But most of the run-of-the-mill services are "automatic." That means that they all start up pretty much the same time as the GUI.

Most of them? Go ahead, count the number of services in Windows. And their size. You can see what Microsoft was thinking: "holy moley! About 40 percent of the stuff that slows down Windows boots are these service things!"

So they created a new class of services: "automatic…but delayed." They get the go-ahead to start up at the same time as the "automatic" services, but they have this special note on them that says "well, yeah, they're automatic, but…take your time." Any of these so-called delayed services start up in quite low priorities. And that can mean that, well, it may take a while for those subscriptions to pop up because, as you've probably guessed by now, winrm and wecsvc are both delayed services.

The bottom line is: don't be surprised if you don't get any updates from the source systems for (in my experience) seven or eight minutes. If that's annoying, you can always bump up the winrm and wecsvc services from "automatic-delayed" to "automatic," although I'd leave them alone. (And consider joining Peter in that switch to decaf.)

Event Forwarding in Workgroups

Before I leave this introduction to event forwarding, I want to provide a more complex example of event forwarding setup because it'll give me a chance to show you some more configuration options and, even better, more troubleshooting tools. First, let's review the steps that we employed to get the two domain members Vista1 and Vista2 to talk, so that we've got a starting point in explaining how to set up forwarding in a workgroup:

- On the source computer Vista2, we started the winrm service so that the source computer could listen to the collector's requests.
- On the source computer, we put the collector's machine account into the source's "Event Log Readers" group so that when the collector asked the source for its events, then the source computer was already configured with permissions allowing the collector to request those events.

- On the collector system Vista1, we started up the Windows Event Collection service; we had to do that because wecsvc does the actual asking for the events from the source.
- On the collector, we configured the subscription. And waited.

The difference in setting up a subscription between two systems that are not members of the same Active Directory forest lies in the security part. We can't just put Vista1's machine account in Vista2 because the two systems don't have any kind of trusting connection between themselves, as they would if they lived in the same AD forest. So we'll have to do some security adjustment.

To try this out, set up two systems named, again, Vista1 and Vista2. They should not be members of the same AD forest. (They *can* be members of domains, but not domains in the same forest or in forests that trust each other.) Ensure that the two computers can resolve vista1.bigfirm.com and vista2.bigfirm.com in DNS properly.

Once that's done, enable the Administrator accounts on both systems and set the password of the Vista1 administrator to "bigpebbles1" and the password for the Vista2 administrator account to "bigpebbles2." Now we're ready to start.

Step One: Configure WS-Management on Vista2

Just as we did before, we'll start by getting WS-Management running on Vista2, the source computer. Log onto Vista2 as its local administrator account. Start up an elevated command prompt and type

```
winrm quickconfig
```

Vista does not apply UAC to the default administrator account, so I don't actually have to explicitly raise the privilege on my command prompt when logged in as "administrator." But Vista can be configured to constrain the default administrator account as well and your system might be configured differently, so I'll keep reminding you to ensure that the command prompts are elevated.

Type **y** and Enter when it asks if you really want to set up WinRM, and you're done on Vista2.

Step Two: Tell the Collector to Trust the Source

Now move over to Vista1 and log on as its local administrator. Because Vista1 and Vista2 no longer share a domain relationship, we're going to have to see how to authenticate between Vista1 and Vista2. Open up an elevated command prompt on Vista1, the collector, and type this:

```
winrm set winrm/config/client @{TrustedHosts="vista2.bigfirm.com"}
```

That tells Vista1 that it's okay to, if necessary, attempt logging onto Vista2. This seems a bit odd to me—you'd think that you'd have to tell Vista2 to trust Vista1, rather than the other way around—but that's how `winrm` works.

That's something of a strange-looking command, so let's take a moment and understand it. `winrm` has a number of configuration parameters that you can examine and change. You can see them all by typing

```
winrm get winrm/config
```

You'll get about a screen's worth of output; here's what some of it looks like:

```
C:\>winrm get winrm/config
Config
    MaxEnvelopeSizekb = 150
    MaxTimeoutms = 60000
    MaxBatchItems = 20
    MaxProviderRequests = 25
    Client
        NetworkDelayms = 5000
        URLPrefix = wsman
        AllowUnencrypted = false
        Auth
            Basic = false
            Digest = true
            Kerberos = true
            Negotiate = true
        DefaultPorts
            HTTP = 80
            HTTPS = 443
        TrustedHosts = vista2.bigfirm.com
...
```

Notice that this information is presented with varying levels of indentation. That's intended to convey the hierarchical nature of these settings, which is referred to as the "WS-management schema." The first line, `Config`, is not indented because everything here is part of `Config`, the configuration information. The next line, `MaxEnvelopeSizekb=150`, clearly expresses that there's a parameter named `MaxEnvelopeSizekb` (and don't ask, because I have no idea what it does) with value "150." But then skip down a few lines to `Client`. That's a level down from `Config` and is a container or, rather, subcontainer of its own. Notice that TrustedHosts is a parameter within Config/Client.

You can use this hierarchy with a `winrm get` or `winrm set` command. For example, the `winrm get winrm/config` command showed the entire schema; to see just the items in config/client then you'd type **`winrm get winrm/config/client`**. That would look like this:

```
C:\>winrm get winrm/config/client
Client
    NetworkDelayms = 5000
```

```
    URLPrefix = wsman
    AllowUnencrypted = false
    Auth
        Basic = false
        Digest = true
        Kerberos = true
        Negotiate = true
    DefaultPorts
        HTTP = 80
        HTTPS = 443
    TrustedHosts = vista2.bigfirm.com
C:\>
```

You've already seen how to set a value in the winrm schema:

```
winrm set winrm/place-in-schema @{parameter="value"}
```

Where *place-in-schema* refers to wherever in the schema the parameter is stored. You can use "*" as a wildcard in the TrustedHosts value, and you can specify more than one item in the TrustedHosts parameter by putting a list of machines in quotes with commas between them. For example, this command would work fine to tell a system to trust both mypc.bigfirm.com and yourpc.bigfirm.com:

```
winrm set winrm/config/client @{TrustedHosts=
"mypc.bigfirm.com,yourpc.bigfirm.com"}
```

That's all one line and do not put spaces between the machine names in the list. (Wondering why I'm spending so much time showing you how to configure and control winrm? Because all kinds of things in Vista and Server 2007 are built atop winrm. Trust me, you'll end up using this stuff.) And while you probably don't want to do this in a production environment, here's an example of trusting everyone:

```
winrm set winrm/config/client @{TrustedHosts= "*"}
```

Reviewing, then, the first thing that we've seen that's new in setting up event forwarding between two systems who lack a domain relationship is that we've got to use the `winrm set` command to add the *source* computer to the TrustedHosts list on the *collector* computer.

Step Three: Test WS-Management Connectivity

This step is not absolutely necessary, but I'm a big believer in a little belt-and-suspenders work up front to avoid pain later. Open an elevated command prompt and let's try a winrm's ping-like command. As before, type

```
winrm id -r:vista2.bigfirm.com
```

This time, however, it won't work. Winrm needs to know how to authenticate between systems. Now, if you don't *tell* it how to authenticate, then it'll just assume that it can use Kerberos, which worked fine when both systems were in the same domain, and that's why this command worked fine in the earlier example between domain members. That's not true any more, so we've got to tell it how to authenticate. The simplest way is to just say, "don't use any authentication." That looks like this:

```
winrm id -r:vista2.bigfirm.com -a:none
```

The option -a: is short for "authenticate." It takes several possible options:

- `-a:none` means "don't authenticate." As the id command is pretty basic, the winrm service is fine with skipping authentication in this case, and the command will work. I mean, we folks with an interest in security are paranoid, but there are limits—I'm not ready for a "secure ping" or "authenticated ping" yet, although I fear that it may happen one day.
- `-a:basic` says to authenticate using a cleartext username and password. This will not work by default, as winrm forbids all unencrypted logon attempts. You can change that with this command:

  ```
  winrm set winrm/config/client @{AllowUnencrypted="true"}
  ```

  But I don't recommend it. You must follow a `-a:basic` command with a username and password that must be written as `-u:`*`domain`*`\`*`username`* `-p:password`. We'll see an example of that soon.
- `-a:digest` uses the same "digest" logon mechanism as the one that Internet Information Services has supported for years. `winrm` considers it an unencrypted logon method and digest logons will not work by default.
- `-a:kerberos` says to use Kerberos authentication. This is the default behavior and, again, requires a domain.
- `-a:negotiate` says to use a Windows NTLM-like challenge-response logon. It's encrypted and will work with winrm's defaults.

To make a negotiated logon work, add the `-u` and `-p` options. As we've set the Vista2 administrator's password to "bigpebbles2," we could make this winrm id work:

```
winrm id -r:vista2.bigfirm.com -a:negotiate -u:vista2\administrator -
p:bigpebbles2
```

(That's all one line, even if it did break on the page.) That should work, and at the same time we've verified that Vista2 can accept logons with its administrator account from Vista1. Armed with that knowledge, we're ready for the next step.

Step Four: Set Up the Subscription

Still working at Vista1, set up the subscription just as we did before, except for two changes. First, you'll have to establish a connection of some sort with Vista2, or the Event Viewer won't

let you even punch in "vista2" as a source computer. My answer was to briefly create an exception in Vista2's firewall for "file and print sharing." Then, from Vista1, I typed

```
net use \\vista2\ipc$ /u:vista2\administrator bigpebbles2
```

That connected me to Vista2. I was then able to specify "vista2" as a source computer.

The second issue also concerned authentication. In the Subscription Properties dialog for the subscription that let Vista1 suck events from Vista2, there's a button labeled "Advanced…" Click it and you'll see a section in the resulting Advanced Subscription Settings dialog box labeled User Account. In there is a radio button Specific User. Click that radio button and you'll see a button labeled "User and Password…" Click it and you'll get a chance to punch in a username and password. Punch in **`vista2\administrator`** for the username and **`bigpebbles2`** for the password. Wait the requisite 15 minutes and you'll have events!

To summarize, setting up a workgroup-based event forwarding differs from a domain-based one in a few ways.

- You've got to add the source computers to the `TrustedHosts` parameter in winrm on the collector computer.
- You've got to temporarily open up file and print sharing on the source computers and then connect in some way—I used the IPC$ trick—to establish a session with those computers. Then it's possible to add the computers to the list of source machines in Subscription Properties.
- In Advanced Subscription Properties, you'll need to punch in the names and passwords of accounts on the source systems who are members of their local Event Log Readers groups.

In this chapter, I took you through a quick tour of some of what I call the Vista "small surprises." In the next chapter, we'll tackle our first big surprise—User Account Control.