Realtime publishers

February 2009
Volume 2, Number 2

Windows Administration
in Realtime

2 ***Letter from the Editor***
Lucky Number 13

3 ***Answers from the Experts***
What's the best way to manage application deployment and removal for onsite contractors?

5 ***10 Linux Commands Every Windows Admin Should Know***
Discover the Great Power of the Command Line
By: David "Dragon" Michaels - Equip yourself with fundamental commands that will help ease your transition from the Windows world and learn about the tools you need to expand your knowledge and improve your comfort with the Linux environment.

15 ***Lessons from the Field***
A Salesman's Perspective of Microsoft Licensing, Partners, and Agreements
By: Jim Varner - Learn Microsoft contract options so that you can quickly determine whether your sales rep really wants to help your organization find the best type of agreement or is just looking to make a sale.

20 ***Practical PowerShell***
Performance Reporting with Windows PowerShell
By: Jeffery Hicks - With a small time investment, you can gain all the performance monitoring and reporting capabilities you need.

30 ***Exclusively Exchange***
Designing for Disaster Recovery
By: J. Peter Bruzzese - Discover the options are at your disposal for recovering from an Exchange disaster.

Letter from the Editor

Lucky Number 13

by Greg Shields

Welcome to our 13th edition of the *Windows Administration in Realtime* eJournal. This lucky-numbered issue is being written on New Years Eve of what some consider the hardest economic situation the United States has seen in decades. The massive devaluation of net worth that resulted from multiple, concurrent incidents including wars, bad property decisions, and massive deregulation have readers like yourself wondering how 2009 will turn out.

Now you may or may not know that magazines like this one are usually written months ahead of time. Production schedules require that articles come in many days before release. So, as I write this letter from the editor for the February issue, I'm about to take off for an evening's celebration of the New Year. This gives me a unique perspective to reflect on 2008, and then later catch up once this issue releases to see how things are going 2 months into 2009.

Whether the economic situation has hit your livelihood or not, my crystal ball tells me that it is likely our current situation hasn't changed in a few short months. Notwithstanding what you read and hear on the cable news networks, the general sense of well-being in this county has taken a nose-dive. And it's likely not to come back for a while yet. With jobs being lost and budgets constricting, there's plenty of reason to think that you're not as well off as you could be.

Yet there is a light at the end of this tunnel. These downturns are but cycles. Until just recently, the IT industry saw massive expansion. Servers, services, and hardware were purchased and brought into production in numbers heretofore unseen. Although not completely recession-proof, the IT industry is one—like power and water—that is needed by essentially every company to keep the wheels of business turning. So, although some of us may lose jobs or find others, those that remain will find themselves with new work to do and greater responsibilities.

To that end, rather than commiserating on how things were or how they should be, take 2009 as an opportunity to expand your skills. Learn a new operating system. We've brought in Dave "Dragon" Michaels in this issue to do just that with the Linux OS. Resign yourself to getting your hand off the mouse and learn to accomplish your necessary tasks through automation tools and scripting. Regular contributor Jeff Hicks helps you every month along that path. Teach yourself a new product or a new way of solving problems that makes you just that much more useful to your business. If you're like me, you'll find that a little personal growth goes far in bolstering the spirits should the economic woes get you down.

Above all, have fun this year. 2009 may not (have) start(ed) off with a bang (now there's a prediction!) but it is the next phase in this business cycle. Your future in IT depends on you making the best of it.

And while you're at it, keep it tuned right here to the *Windows Administration in Realtime* eJournal where we're giving you all the very best information to help!

Answers from the Experts

What's the best way to manage application deployment and removal for onsite contractors?

by Don Jones

Q: This month, Allan asks about software deployment: "We have a number of contractors who come on-site and join their computers to our domain. We need to be able to deploy an internal application via Group Policy but want the application to go away after we remove the contractors from the domain. What's the best way to do this?"

A: It's a great question. In fact, Group Policy would seem to offer a built-in solution: When you deploy a package using a GPO, you have an option to remove the package once the person or computer is removed from the management scope. Typically, this is used when someone in a given OU—say, a Sales OU—is affected by a GPO that deploys a Sales-specific application to their computer. Remove them from the Sales OU, and that application will be removed automatically. However, the trick doesn't work as reliably when you're removing someone from the domain entirely.

A better technique that's more reliable uses Microsoft's App-V (formerly SoftGrid) technology. In fact, once you get to know App-V, you might want to start using it for

all your application deployments. Technically, Microsoft calls it *application virtualization*, although I think the word "virtualization" is getting a bit overused these days; I prefer to call it *application sandboxing*. Essentially, you (the admin) use App-V's management tools to install an application into an App-V sandbox. This installation process is called *sequencing* the application.

You can then use a variety of techniques to deploy the application, from a simple file copy to a fully managed deployment using App-V's "full infrastructure" mode, which requires additional management servers on your network. Users don't install the app; they simply get a copy of the sandbox and a copy of the App-V client software. The client knows how to get the sandbox up and running—with the application inside already installed and ready to run.

The neat thing is that the sandbox (or *package*, to use the correct term) can be configured with management metadata—such as an expiration date. The App-V client can automatically remove the package after that date passes, and there are a number of other management criteria you can specify to manage the app's lifetime. The bonus part about the app being sandboxed is that once the package is removed, nothing from the application remains on the computer: no registry keys, no buried files and folders, *nothing*. This "zero footprint" deployment means that removing the application is literally as easy as deleting a file.

The major downside to App-V is that it's only available to Microsoft customers who have purchased Software Assurance. Frankly, I think that's a somewhat bad decision on Microsoft's part. It was bundled that way primarily to give value to all the customers who bought Software Assurance on Windows XP and then essentially got jilted out of the free OS upgrade they were expecting, due to Vista's long development cycle. Today, App-V and some of the other applications in the Microsoft Desktop Optimization Pack (MDOP includes software besides App-V and is only available to Software Assurance customers) are probably the driving reason to spend the extra money on Software Assurance—if you're eligible for it. The main reason I don't like everything being tied to Software Assurance is that most smaller companies without Volume License agreements can't get Software Assurance. Allan, you didn't mention what size company you're with, but if App-V is an option for you, it should be exactly what you need.

Don Jones is a co-founder of Concentrated Technology. Join him and cohort Greg Shields for intense Win2008 and Windows PowerShell training—visit ConcentratedTech.com/class for more details. Ask Don a question by visiting ConcentratedTech.com and using the "Contact" page.

10 Linux Commands Every Windows Admin Should Know

by David "Dragon" Michaels

So you, a Windows veteran, find yourself staring at a text window that says only ">". For some reason, you've gone through the trouble of getting yourself an account on a Linux or UNIX box. You're probably getting to it remotely via Windows telnet or more likely an SSH client like PuTTy. Or maybe you're on the console itself, marveling at the server's window manager that looks more or less like Explorer and you've launched yourself a terminal window.

Congratulations! You have entered the brave new and ancient world of the Linux command-line interface. It may seem archaic at first, like a relic of a time gone past. That feeling will probably never really go away, but within this very command line lies great power. It's a lot easier to write scripts using commands than it is using a GUI. And since commands themselves are easier for coders to create in the first place, there are thousands of them available; what they lack in ease of use they make up for in capability.

To get the most out of this article, you need to know only the most basic of Linux commands, like `ls` and `cd`. The article will introduce you to 10 other Linux commands that you might find useful. Most will work the same in all flavors of Linux, while one or two are used specifically in Red Hat Linux. Although Linux and UNIX are sometimes considered distinct, they're virtually identical when it comes to command-line stuff, so most of these tips will be just as useful for a UNIX user.

#1: man

The first command for this article is man, which is the first command most Linux admins ever learn. `man` is the command you use to bring up a "man"ual page for virtually any command. It also provides info about library calls, file formats, and many other aspects of the system.

`man man` is the key to unlocking the secrets of all other Linux commands. This command tells you how to use the `man` command itself and is one that even Linux masters never stop using. At the end of most "man pages" is a section titled "See Also," which will lead you to other commands you might find useful or interesting.

You've probably used Windows Help before, and you may have found that Help is hit or miss on usefulness. Linux man pages are no different; they are infamous for being cryptic and difficult to decipher. However, they have improved over time, so it's worth the effort to become familiar with them.

Type `man man` at your Linux command prompt, and you'll see the command itself has many options, including keyword searches, searching non-standard paths, and converting to other outputs for printing. You can even type `man intro` for a brief introduction to Linux.

`man` automatically uses something called a pager that provides the mechanism for easily navigating its pages. The default pager is usually `more` and is available on every type of Linux. When using `more` to read a file, hit the space key to see the next page, "b" to go back a page, and "q" to quit. There are often alternative pagers with more capabilities than `more`, such as the pager called, well, `less`. Which brings new meaning to the phrase, "less is more." Linux is not without a sense of humor.

Next up is the invisible Linux command, the shell itself.

#2: the shell (tcsh, bash,…)

Early on, the original legendary UNIX programmers built themselves a program called a "shell." The shell interprets any command you type, and generally tries to make things easier for you. In Windows, `cmd.exe` is a shell, and `explorer.exe` is a graphics shell. A shell can also run a file with a series of commands, which is called a shell script. These work much the same as a batch or even PowerShell file in Windows. In Linux, a file is a shell script if it begins with #! and is executable (for example, a bourne shell script starts with #!/bin/sh, which instructs the computer to use /bin/sh to interpret the file). You can make your own scripts this way. Just be sure to type `chmod u+x filename` to turn the otherwise normal text file into an executable.

Some shells are great for shell scripts but terrible in terms of interactivity. Some are the other way around, and still others (such as bash, the "bourne again shell") are good for both.

The most important thing to know about shells is how to tell them to break out of a command. To do so, type Control-C, which will attempt to terminate the command. Alternatively, you can type Control-Z, which will suspend the command. To bring it back after doing so, enter `fg` for foreground. In Linux, this is called job control.

One logon shell you might start with is tcsh. Type `man tcsh`, and you'll see that it is rich with features. This makes it a fantastic login shell but a terrible scripting shell. Type `echo $0` to see what shell you're using. If it's not /bin/tcsh, you can type `/bin/tcsh` now to change it temporarily and follow along.

Your shell's prompt is also important. Linux is much easier to use if your prompt helps keep track of where you are and what you're doing. Thus, setting a friendly prompt is the first thing you should do with a new Linux account. As an example, in tcsh, type:

```
set prompt="(%?) %T %n@%m <%c2><%h%# "
```

Which results in this prompt:

```
(0) 20:02 user1@server1 <~/Directory1><3>
```

This shows a lot of information at a glance:

- The (0) indicates the "exit code" of the last run command. In Linux shells, 0 is considered a successful result, while any other response is not.
- The middle part tells me the time, and who and where I am. That becomes much more important if you have multiple accounts or multiple Linux boxes, and it's critical if you have root. Knowing whether you're running a command as a regular user or as the root user could mean the difference between a benign error and massive disaster.
- The <~/Directory1> part is the current directory. ~ is shorthand for "my home directory" in most Linux shells
- Last is <3>, which shows the current command number. In tcsh, you can type !*number* to repeat the Nth command you entered, and you can type `history` to see them all. For example, !2 will repeat the 2nd command you typed. In most shells, you can use the arrow keys to edit command lines and navigate up and down through the command history.

tcsh has a useful feature called filename completion. When entering a filename, type a few characters, then hit the TAB key. tcsh will fill in the rest.

This is a good time to introduce a special flag you can pass to most Linux commands, the --help flag. For example, type "smbclient --help", and you will see a short summary of all the possible options for the smbclient command. If you don't have a scrollbar, you can pipe the result to the more command. A pipe is a pathway between two commands, connecting the output of one to the input of the next. This also works in Windows cmd.exe. You can chain many commands together using pipes, but in this case, just type this:

```
smbclient --help | more
```

With this little trick in mind, let's enter the wonderful world of Samba with the command smbclient.

#3: smbclient

Samba is a suite of commands that Linux uses to interface with Windows servers. It gives Linux the ability to act as a Windows file server or even domain controller, and allows it to access Windows shares. As a Windows veteran on a Linux box, you'll probably want to familiarize yourself with Samba's capabilities and commands.

Samba itself comes equipped with far too many commands to cover here, so this section will focus on the very simple `smbclient`. You use this command to browse and connect to Windows file shares and to transfer files to and from those shares.

To list the Windows shares on the host \\server1, type:

```
smbclient -L //server1
```

Notice the direction of the slashes in that previous command. Linux uses backslashes for escaping special characters that your shell would otherwise try to interpret. So, instead of \\server1, it's //server1. The previous command will ask you for the Windows password for the account name you're using to run the command. If you need to authenticate as an alternate user, use the –U flag to specify the user. Though Linux is still limited to 8-character account names, smbclient is not. To connect to server1's share called temp, simply type smbclient //server1/temp, and you'll get something like this:

```
(0) 13:21 user1@server2 <~><8> smbclient //server1/temp
Password:
Domain=[SERVER1] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
smb: \>
```

The `smbclient` command is like a mini shell. In this shell, you have a whole new set of commands. Type "?" to get a list of them. To learn more about a particular command, type `? command`. You can list files with `dir` or `ls`, and `get` or `put` files to upload or download them. There are a few dozen other commands, so feel free to explore.

Other Samba commands of interest are `smbutil`, `smbstatus` (for Samba servers), and smbtree. Use their --help flag for more info and use man to learn what they do. With tcsh, if you type smb<tab>, you'll get a list of all the smb* commands as it tries to complete the command name you typed.

Next up: Find anything you're looking for with `find`.

#4: find

Even the most seasoned of systems administrators, Windows and Linux alike, need to sift through piles of files looking for a text string we're sure is there somewhere. We just can't remember where it is. We might even be hunting for something we've never seen before, but something tells us, "It must be here somewhere." Enter Linux's `find` command.

Suppose you can't remember where that darn Network Time Protocol config file is located. You think it has the word "ntp" in the filename, but you're not sure where it lives. This is where find is indispensable, and it's one of the most basic things `find` can do. If you want to find that lost NTP file, you can use:

```
find /etc -name '*ntp*' -type f
```

In this command, `find` will start its search from the /etc directory—where most config files live—and will look for a file whose name contains the word ntp. It's important here to enclose this string in single quotes instead of double quotes.

Searching for things other than filenames can be done by changing the flags passed to the command. The flag `-name` tells `find` to check names of things, and `-type f` makes it check only files, ignoring directories, symbolic links, and other types of things you find on the file system.

A `find` flag that is often useful for systems administrators is -mtime. This can be used to find things that have recently changed or things that haven't changed in a while, both of which can sometimes be quite helpful when the system has recently started acting strangely. For example:

```
find /etc ! -type d -mtime -5d -mtime +1d -ls
```

Would show you everything in /etc that is between 1 and 5 days old, not of type "d" for directory. The ! is pronounced "bang" and is often used in Linux to mean "not." Instead of just printing the name of the file, which is the default behavior, the `-ls` tells `find` to show you the "long listing." For a test to succeed, the things `find` tests must match all parameters. In this case, `find` will list non-directories less than 5 days old and more than 1 day old.

To compare, Windows has a `find` command as well. But it's less like Linux's powerful `find` command and more like a weak version of the Linux command `grep`. Open up a `cmd.exe` window and type `ipconfig /? | find "renew"` to display just the lines from ipconfig that include the word renew.

Coming up next is the fantastically useful troubleshooting tool `top`.

#5: top

You've probably used Task Manager or `pslist -s` from the Sysinternals' PsTools, You'll find the Linux `top` command to be quite similar. `top` shows information about the processes running on the system and the system itself, but in a streaming manner like Task Manager. The following sample is the output from my OSX laptop. The results from `top` from Linux and other UNIXes show the same information in much the same order but with a different layout.

```
Processes: 60 total, 3 running, 57 sleeping... 252 threads     18:52:24
Load Avg: 0.39, 0.45, 0.41  CPU usage: 19.14% user, 11.48% sys, 69.38% idle
SharedLibs: num =  2, resident =  62M code, 6868K data, 4276K linkedit.
MemRegions: num = 9977, resident = 519M +  10M private, 212M shared.
PhysMem: 248M wired, 823M active, 131M inactive, 1210M used, 838M free.
VM: 8736M + 371M  152677(0) pageins, 77603(0) pageouts

 PID COMMAND   %CPU  TIME  #TH #PRTS #MREGS RPRVT RSHRD RSIZE VSIZE
 356 webbrowser 39.4% 4:36:30 25  291  2596 225M  36M  373M  848M
 2225 top     5.5% 0:00.48  1  18   29 972K  188K 1568K  18M
 1831 LaunchCFMA  4.4% 13:53.81 10  201  710 107M  51M  158M  785M
```

This result looks cryptic at first glance, but it's really not much different than what you commonly see in Task Manager's Performance tab. Let's break it down:

- The first line is self-explanatory—60 processes, 57 doing nothing, 3 active. In my case, those would be `top`, a Web browser, and LaunchCFMA (which is really Microsoft Word).
- Next, there's the Load Average, followed by some CPU info. Load average is something you won't find in Task Manager. It is a statistical analysis of the tasks the computer is running, where the numbers correspond to the computer's average load rated by number of jobs in the run queue over the past 1, 5, and 15 minutes. This data is useful if a user-reported problem went away by the time you ran `top`. If the second or third load average number is high, you know that the user's problem actually happened at some point in the short-term past.
- The next few lines contain extensive information about the memory usage on the system, similar to the bottom part of Task Manager. Linux `top` shows total memory, and though OSX doesn't mention total memory explicitly in the PhysMem line, you can tell that you have 2GB of RAM by the sum of "used" and "free" values.
- Finally, you'll see the individual process information. There you can see that the Web browser is consuming almost 40% of the available CPU and is using 373MB of RAM, as shown by the RSIZE field—time to close some You Tube tabs.

Next, you'll learn about system logs and `dmesg`.

#6: dmesg and syslog

One of the most common tools you use for problem analysis is probably the Windows Event Logger. It can be a pretty useful graphical tool, but the Linux world was built around the command-line, so it instead has the text-based `dmesg`.

`dmesg` displays the last set of system messages. If you run this command shortly after a reboot, it will display kernel boot messages. In it, you will see extreme levels of detail about various devices, drivers, and other kernel-level initializations. The output can be fairly verbose, more so than the messages you see when you boot Windows in debug mode. The information provided by each tool is functionally the same—they are designed to show you what the kernel is doing.

In Linux, `dmesg` provides a glimpse into a 16KB slice of kernel memory reserved for such messages. When that space has been filled up with messages, the log loops and new messages are written to the beginning, which overwrites earlier messages. This is important because although `dmesg` shows you everything in chronological order, if your system has been up for a while, the boot messages may have been lost.

To assist with this looping situation, Linux also has the syslog. Syslog is a Linux service that begins its life early in the boot process, and its role is to funnel messages into long-term storage from other services or applications. Invariably, these files are found somewhere in /var, a common Linux directory tree used to store "variable" data. Lots of things live in /var, like the system logs in /var/log, crash dumps, Web server access logs, database caches, lock files, process ID (PID) files, Linux's scheduled tasks called crontabs, and others.

System logs contain sensitive data, as they provide a detailed insight into the system, so you might find that many directories in /var are protected such that only the user syslog or root has access to read or write. For those that you can see, syslog files typically have a format similar to the following:

```
Dec 14 17:00:00 server1 newsyslog[62807]: logfile turned over due to size>100K
```

Specifically we see a timestamp, which often does not include the year, the hostname, the service the message is for (with the PID in brackets), and the message itself. If you ever find yourself building or supporting Linux servers, you'll be looking at these files quite a bit, probably about as often as you check out your Event Log in Windows.

As we're on the subject of boot messages, next on the list is the `init` command, which boots the system.

#7: init

You might have heard the rumor that Linux machines are so stable that they don't even need a reboot command. And you might have believed it. After all, how could such an archaic OS with a clumsy interface be so pervasive if it didn't have something going for it? Well, of course Linux has a reboot command. And for most Linux and UNIX distributions, it's called `init`.

Right about now, you're probably wondering why it's not called reboot or shutdown. Linux commands and files are so intuitive that surely they would have thought to make such obvious commands. Right? In truth, those commands do exist, and they do what you think they might, but `init` is the mother of all Linux commands. Literally.

If you were to view the processes on just about any Linux system, you'll see that the init process is the first in the list:

```
22:03 (0) user1@server1<~><143> ps -ef | grep init
root     1   0 0 Dec07 ?    00:00:00 /sbin/init
```

This means that `init` is the first process the OS creates, and is thereby the parent to every other process—the mother of them all. If `init` doesn't start your process, it starts something that starts your process, all the way down to the last process on the system. But `init` is also a command you can use to instruct that `init` process to do things to the system such as shut down or reboot.

To understand this further, you need to understand the different states in which a Linux system can run. Usually, Linux runs in something called "runlevel 5". Think of a runlevel as a preconfigured boot profile that describes how the system will be used. In runlevel 5, `init` brings up services that support a GUI console, a network connection, and network-related services. There are other system runlevels as well. Runlevel 3 is like runlevel 5, but has no GUI console. Runlevel 0 is the shutdown state or halt state, and runlevel 1 is the single-user state.

When you instruct the init process to change runlevels, it runs a series of scripts to bring up or take down that runlevel's associated services. To shut down the system, you tell init to move to runlevel 0 by typing `init 0`. The system will then gracefully shut down all its running services in a particular order. When everything has fully terminated, it will then terminate itself and thus stop the system. If you want the system to come back up after it shuts down, use "init 6" instead. The `shutdown` and `reboot` commands are really just front-ends to init 0, with a few more options.

Now that you know all about `init` states and runlevels, next up is a Linux command you can use to customize them called `chkconfig`.

#8: chkconfig

Probably the biggest difference among the various Linuxes and UNIXes out there is the manner in which they handle a system boot. Each performs this operation differently. The one thing that is consistent across each is the presence and purpose of the `init` process. Use this fact to your advantage and begin learning about your OS' boot process by typing `man init`.

When it starts, the first thing `init` does is parse the /etc/inittab file. You can query man for information about file formats, which in this case would be `man inittab`. Your inittab file may attempt to configure many things—depending on how industrious your Linux administrator is—but in Linux, there are always lines that call to scripts that are found at /etc/rc.d/rc. This script is the engine for starting and stopping services in a particular order for the various runlevels. You manage this engine with `chkconfig`.

Let's look at how the `chkconfig --list` command works. The following example shows the first two lines that appear when running this command.

```
16:04 (0) user1@server1 <~><2$ /sbin/chkconfig -list
ConsoleKit   0:off  1:off  2:off  3:on  4:on  5:on  6:off
NetworkManager 0:off  1:off  2:off  3:off  4:off  5:off  6:off
```

The first column lists service names. Subsequent columns show you whether that service is enabled or disabled for that runlevel. In this case, ConsoleKit starts in levels 3, 4, and 5, but NetworkManager is fully disabled. To see what services are different between runlevel 3 and runlevel 5, you can pipe the result through a couple of grep commands:

```
16:02 (0) user1@server1 <~><1$ /sbin/chkconfig --list | grep '3:on' | grep '5:off'
gpm       0:off  1:off  2:on  3:on  4:on  5:off  6:off
16:04 (0) user1@server1 <~><2$ /sbin/chkconfig --list | grep '3:off' | grep '5:on'
readahead_later 0:off  1:off  2:off  3:off  4:off  5:on  6:off
```

In these results, you quickly see how gpm does not run in runlevel 5 but does run in runlevel 3, while readahead_later does just the opposite.

It was discussed earlier how the GUI typically is run in runlevel 5 but not runlevel 3. But neither of these two services sound like the GUI, so how does this work? The answer lies in /etc/inittab, in a spot beyond the "rc script" that `chkconfig` manages. The GUI engine is called X11, which you should find reference to somewhere within that file. X11 is effectively the starting point for the Linux GUI.

The fact that the gpm service is not running in runlevel 5 presents another excellent example of where you can make changes to your system to make it easier to administer. The gpm service is great for virtual consoles—it allows you to use a mouse for cut and paste on a text console. So you might want enable it in runlevel 5 as well because virtual consoles are still available outside the GUI. In the following example, you'll see where three commands are run to show how gpm begins as disabled in runlevel 5. It is then configured to run there in the second command, and then verified by re-running the `--list` command.

```
16:57 (0) root@server1 <~><5# chkconfig --list /sbin/gpm
gpm       0:off  1:off  2:on  3:on  4:on  5:off  6:off
17:01 (0) root@server1 <~><6# chkconfig --level 5 /sbin/gpm on
16:57 (0) root@server1 <~><5# chkconfig --list /sbin/gpm
gpm       0:off  1:off  2:on  3:on  4:on  5:on  6:off
```

Be aware that any changes to services like the previous example must be done as root. You can tell by the prompt that the root user is the user running these commands. Between `chkconfig` and modifications to /etc/inittab, you can completely customize how your Linux box behaves on boot up and shut down, and you can define what each of the runlevels mean. You can find detailed customization guides online; for example, check out this one from *Red Hat Magazine*.

Coming up next is `rpm`, a tool for managing installed packages.

#9: rpm

Let's take a moment to recap what you've learned. You now know about the boot process, runlevels, and how to turn on and off services. You know how to look at processes and find stuff on the system. You know how to make Linux talk to Windows, how to talk to Linux via the shell, and how to learn about all the commands Linux has.

You may be thinking, "But what about the commands and applications I know are out there but my Linux doesn't have them? I know they're missing because I can't find them with `find`!" Well, put that `find` away, because you can find and manage your packages a lot more easily and quickly with `rpm`.

`rpm` was originally Red Hat's Package Manager, a special bundling format for software packages. It is now freely available and open source so that all vendors can use the same format to distribute their packages as well. You'll see this kind of thing often in Linux where companies reuse and share discoveries and developments. Some examples are NFS, NIS, Java, and lately ZFS, all of which originally came from Sun Microsystems but today are ubiquitous across Linux and other platforms.

As a regular user, there is not a great deal you can do with rpm because it typically requires root to install, remove, or otherwise modify system packages. But you can query what's out there. To see everything installed on your system that rpm knows about, type r`pm --query --all`, and be prepared for lots and lots of output.

```
19:48 (1) user1@server1 <~><26$ rpm --query --all
comps-extras-12-1
atk-1.18.0-1.fc7
libjpeg-6b-37
…
[snip]
…
git-svn-1.5.2.1-1.fc7
zenity-2.18.2-1.fc7
```

You'll notice that they're not in any particular order, but you can pipe the results through the `sort` command to fix that if you desire. Using the `rpm` command alone, you could get a reasonably good snapshot of your system's software baseline, which can be a useful troubleshooting tool when comparing a working system to a malfunctioning one.

Another good use for `rpm` is to find out what package owns a particular file of interest. Say you come across /usr/bin/cjpeg, and you want to know more about it. Just ask `rpm`. While the --file option expects a file and returns the package that owns that file, the `--list` option expects a package and reports the files owned by that package.

```
20:01 (0) user1@server1 <~><38$ rpm --query --file /usr/bin/cjpeg
libjpeg-6b-37
```

You can use `rpm` to install packages if you have root using the `--install` option. Most `rpm` files you download—referred to simply as rpms—will have signatures you can use to verify their authenticity, and the rpm command supports this nicely with `--checksig`. I recommend doing this for any rpms you install, although in many cases it is automatic.

```
20:54 (0) user1@server1 <~><55$ rpm --checksig ImageMagick.src.rpm
ImageMagick.src.rpm: sha1 md5 OK
```

This command returns an OK upon running to show that the package's MD5 checksum—and therefore its authenticity—have been verified.

Not all UNIXes have `rpm`, and some UNIXes, such as AIX, have `rpm` as an afterthought in addition to their native package manager. Thankfully, someone created a wonderful tool for translating commands from one UNIX or Linux to the next, called the Rosetta Stone of Unix. Whether you're new to UNIX or Linux or an old hat at all of them, there's something this tool can teach you.

Now that you have more confidence in your Linux skill set, it's time to give you one of the most powerful—and potentially dangerous—tools in your Linux quiver: the text editor `vi`.

#10: vi

Because Linux and all UNIXes are based almost exclusively on configurations stored in text files, the ability to edit them without the knowledge of the potential impact can be quite dangerous. There was a time in this writer's experience where a stray ':' in a host database file brought an entire network down. Most Linux files of this power that are so sensitive are restricted to root-only access. So if you have such access on your machine, be very careful. Make copies of any files you tweak before you tweak them. Even as a regular user, you could render your account unusable, forcing you to humble yourself before your local Linux administrator.

`vi`, pronounced "vee eye," is the most ubiquitous text editor across all flavors of UNIX, written more than 30 years ago by Bill Joy. As with many Linux commands, you invoke it by providing it a filename to work with. For example, `vi ~/.tcshrc`.

Doing so will replace your text screen with the `vi` interface and open your .tschrc file. Alternatively, you can use the `view` command, which will open the file in read-only mode. An important thing to remember about `vi` is that it has two main modes: navigation and editing. It always starts in navigation mode, where your arrow keys should work fine. There are several ways to enter editing mode:

- "i" will go into "insert mode," and begin inserting text at the cursor
- "a" will go into "append mode," and append to the end of the current line
- "x" will delete the character at the cursor
- "d" is a prefix command for different kinds of deletes; "dw" takes you from the cursor to the end of the word the cursor is on, "d3w" deletes 3 words, "dd" deletes the line, and so on
- <ESC> will exit editing mode and return you to navigation mode

While in navigation mode, typing a ":" will let you enter commands. There are many commands available, including ways to create macros, save, quit, and so forth. Consider these useful commands:

- :help—brings up a help interface
- :q—quits the program and prompts you to save changes
- :q!—quits the program without prompting
- :w—writes (saves) the changes to the file and lets you continue editing
- :wq—writes changes and quits

It is becoming more common these days, particularly on open source Linuxes, to direct the `vi` command to invoke `vim`, Vi IMproved. To find out if you have `vim` installed, use `rpm`, or just type `vim`. If you have `vim`, you should also have a command called `vimtutor`, which uses the editor itself to train you how to use it. You will want to spend a good amount of time learning to use this editor.

If you still find `vi` a bit clumsy, `emacs` is a good alternative, as it shares more similarity with Windows Notepad than vi does. Type `emacs -f help-with-tutorial` to learn how to use it.

Summary

As you can probably tell by now, there is a great deal of information about these 10 Linux commands this article could not cover, to say nothing of the myriad other commands at your disposal. You are now equipped with fundamental commands that will help ease your transition from the Windows world while giving you the tools you need to expand your knowledge and improve your comfort with the Linux environment. ◆

David "Dragon" Michaels has 19 years of professional experience in the IT industry working as a systems administrator, engineer, and architect in multi-platform environments. Though his emphasis has been on the many flavors of UNIX and Linux, he has also worked extensively with Windows as well as Windows to UNIX integration. He has a B.S. in Computer Science and a B.S. in Environmental Engineering from New Mexico Tech, and has been certified with Red Hat Linux and Solaris. Dragon currently works in IT for Raytheon in Aurora, Colorado.

A Salesman's Perspective of Microsoft Licensing, Partners, and Agreements

by Jim Varner

Microsoft software is on desktops and servers, it's at the core of many network appliances, and is even showing up in your car. So how does a company of this size, with intellectual property that spans the globe, create a purchasing mechanism that makes everyone happy? The answer is simple—it doesn't!

With literally millions of users worldwide, it is impossible to create a selling model that will fit every situation. However, if you examine the purchasing vehicles that are in place and consider the enormity of the task at hand, you have to hand it to Microsoft's contract development teams. Microsoft licensing contracts maximize the user's buying power and offer flexible terms that make the life of the admin easier. However, to take full advantage of all the utilities and services available, you must spend a good deal of time learning about the available benefits.

This article outlines some of the different Microsoft contracts and explains the basics of how they work. It then delves into Microsoft Software Assurance and illustrates how it is more than just simple upgrade protection. Finally, we will tackle the problem of how you can find a partner with an in-depth understanding of how Microsoft contracts work and the ability to help you choose which type of agreement will benefit your organization most. There are a few tips and tricks available that can help you quickly determine whether your sales rep really wants to help you succeed or is just looking to make the short-term sale.

Licensing Basics

If you remove the OEM and FPP licensing programs from the mix, all the remaining contracts fall under the Microsoft "Volume Licensing" program. Microsoft publishes a list of all software packages that are available through the Volume License Program here. The most common types of agreements are Open, Select, and Enterprise. Although some simple distinctions can be made (such as the number of licenses purchased), it is important to understand the basics of how these programs work.

The Open License and Open Value programs offer perpetual software licenses. Perpetual licenses are licenses that you "own" as long as you follow the terms of the agreement. There is also a third type of Open agreement, Open Value Subscription. The Subscription contract offers licenses with a limited lifespan. When your Open Value Subscription contract expires, you no longer have the right to use the software. The Open program is designed to fit companies that have anywhere from 5 to 250 PCs. Open Licensing offers discounted prices as well as flexible payment options (Open Value), and 2- or 3-year terms. It should be noted that not all licensing programs include media with the first purchase. Thus, you might need to purchase or download media kits to install the software. The main difference between the Open License and Open Value programs has to do with Software Assurance. Open Value requires you to purchase the Software Assurance when you acquire your licenses, and the Open License program makes the Software Assurance optional. The Open License programs are ideal for small businesses that have a variety of Microsoft software packages and want to leverage their buying power across platforms. Midsized businesses can also benefit from this arrangement if they can't reach the minimum purchase requirements for a Microsoft Select Agreement.

Microsoft Select programs are targeted at organizations with 250 or more desktop PCs. There are two types of Select Agreements: Select License and Select Plus. An unusual trait of the Select program is that it divides your software purchases into three pools: Application, Server, and Systems. In order to qualify for this pricing, you need a minimum of 1500 points in any one pool (Windows Server 2008 Standard is 15 points). Once you qualify, the discounts improve with quantity based on the following scale:

	Select Agreement	**Select Plus Agreement**
Level A	1500 to 11,999 points	500 to 3999 points
Level B	12,000 to 29,999 points	4000 to 9999 points
Level C	30,000 to 74,999 points	10,000 to 24,999 points
Level D	75,000 or more points	25,000 or more points

The Select License program allows you to forecast the amount of points you will purchase over a 3-year period, and purchase licenses at that point level. The Select Plus program has no forecasting component but your discounts are not applied to your pricing until you hit the number of points required in any of the three pools over a 1-year period. Both programs allow you to purchase licenses with or without Software Assurance and have flexible payment terms. If you have an active Select Agreement and meet the correct criteria, you can "roll" Microsoft Select into a Microsoft Enterprise Agreement.

Microsoft Enterprise Agreements are designed for organizations with 250 or more PCs. These agreements scale well into very large corporations, as they require a centralized purchasing model and require you to standardize on specific Microsoft applications across the entire organization. With this level of commitment, you will be in a position to take advantage of the tools and utilities Microsoft offers. These tools will monitor your systems and move your organization towards a standards-based environment where the software on your desktops is merely a utility for delivering information to end users.

Because they are targeted at large organizations, Enterprise Agreements are focused at a level above providing code that runs your hardware. There is an entire division at Microsoft called the Enterprise and Partner group that is devoted to "delivering solutions that secure and advance the business and amplify the contributions of people." By shifting the focus off of software as code, Microsoft is attempting to integrate your company's business and productivity into its offerings. Although this is true at all levels, with an Enterprise Agreement, you will be close enough to see this benefit through the tools and utilities you have at your disposal. This business focus is one of the main reasons that Software Assurance is mandatory if you have an Enterprise Agreement.

Software Assurance

Most people view Software Assurance as the right to download the latest version of the software at no charge while the contract is in place. Although this is true, there are many other benefits that are bundled with the purchase of Software Assurance. Software Assurance packages offer home use programs and training vouchers, and some offer a limited number of free phone incidents. There is a very well-done chart that illustrates all the different "perks" offered by Software Assurance and which Licensing Programs include them. The interactive chart can be downloaded from this Web site. Software Assurance benefits are different depending on which licensing program you choose, but the benefits are significant.

At the most basic level, almost all Software Assurance allows you to spread the payments for your software out over the term of the contract. This can impact your cash flow significantly, and transform your software from a capital expenditure that drains your resources, to a budgeted expense that is more like a utility bill. Another piece of the Software Assurance portfolio that can be extremely valuable is the Microsoft Desktop Optimization Pack (MDOP). This package provides features such as Application Virtualization (ala Softricity) for which third-party vendors charge a significant fee. MSDOP also offers asset inventory tools, diagnostic and recovery tools, and desktop error monitoring. I have spoken with many IT people who manage agreements from basic Open License to Enterprise, and the level of benefit they received from both their Software Assurance and their entire contract, was highly dependant on one simple factor: who they purchased from.

Choosing the Best Partner

At the highest level, Microsoft Partners can be divided into three groups: Value Added Resellers (VAR), Large Account Resellers (LAR), and Enterprise Software Advisors (ESA). All three of these partner levels are focused on providing a different type of service. This section will examine all three partner types to attempt to determine which partner will excel at serving your organization. Then we will dig into simple behaviors to look for as you search for the right match.

VARs usually view Microsoft as a utility. This type of business delivers solutions from multiple software vendors whose platforms require Microsoft "under the hood." One of the people I have had frequent discussions with is an IT director for a dynamic printing and marketing company. His business requires very advanced desktop and Web publishing applications, which Microsoft simply doesn't offer. The organization is not large enough to benefit from the more advanced agreements, and his primary software vendor provides Microsoft support as part of a turnkey solution. In this case, the VAR is a perfect partner. They know about the other software packages being used and understand how Microsoft provides the underlying platform. As a result, they can deliver basic knowledge of what software he needs and how to best acquire the licenses. However, once his organization grows to a point where the office management side of the business requires more advanced features and IT management, he will most likely investigate a Microsoft LAR.

The LAR has a business practice built around maximizing the benefit you get from your Microsoft purchases. Most LARs have advanced license management tools and an in-depth knowledge of the resources available from Microsoft. Typically, a LAR will be able to tell you whether the challenges you are facing can be addressed by the advanced features in the Microsoft portfolio. In the past, I have worked with a large law firm that was considering a third-party software package and consulting engagement to help roll out a new version of Windows. After a brief discussion, they discovered that Desktop Deployment Planning Services is part of the Microsoft Software Assurance package that they had already purchased. As a result, they were able to rollout their new OS without spending more on third-party products. A competent LAR will have detailed knowledge of Microsoft contract benefits and be able to guide you through the steps required to make these programs work for you. This level of service does require a commitment to Microsoft products, but does not require the same degree of standardization necessary to execute a Microsoft Enterprise Agreement.

Microsoft ESAs devote themselves to managing the complexities of large licensing agreements. Intellectual property and the legal agreements that govern them are so detailed that this level of reseller will frequently have legal advisors on staff to explain how an Enterprise Contract will impact your business. ESAs will also have technical teams who are capable of running free audits to ensure compliance and know how to remediate usage problems that may arise. One of my contacts in the energy industry uses an ESA to manage his global user base of more than 6000 users. For him, the most valuable service his ESA frequently provides is taking the results of his quarterly, corporate-wide audits and explaining where there may be compliance and use issues and why. He confessed that his ESA has spent days at a time with him going over these reports to produce budgets and justifications for the CIO. However, without that ESA helping him through the process, it would most likely never be done. Although this level of assistance does benefit a large organization, it would not be helpful to a smaller business who views Microsoft simply as a delivery platform.

Regardless of what size business you have, or what type of partner can best assist you, there are some basic criteria that they all should meet in order to be of any benefit. Any experienced IT manager who has had a valuable relationship with a salesperson knows that the best salespeople ask a lot of questions, and the one you almost never hear is "What would you like to order?" A sales professional will want to learn about the challenges you face, then use whatever knowledge, tools, and resources are available to provide you with definitive answers about how to address those issues. So how do you determine whether your partner is really saving you time and money or simply looking to take orders? A good metric in this situation is how much time you have to spend researching the products you are purchasing and then determining whether you saved time because you made a phone call or sent an email.

Regardless of the type of partner you work with, a basic knowledge of Microsoft and its offerings is required. The following list highlights questions for a potential partner and offers a brief summary of what you should consider when listening to the answer. Hopefully, this will help you quickly determine whether you are talking to the right person:

- **What is Software Assurance?**—This one is fairly straightforward and can quickly tell you what type of partner you are working with, regardless of how they are "labeled" (VAR/LAR/ESA). If you are looking for a basic fulfillment partner, they will tell you about not having to purchase the next upgrade and stop there. More advanced salespeople will briefly explain the program and a few of the features, and the best will give you a few details and ask a question about your company or your job to try to figure out which program will be most beneficial.
- **Can I use the software on my PC at home?**—This question is designed to determine two basic pieces of information. First, to see whether your salesperson understands that Home Use Rights are tied to Software Assurance, and second to dig a little deeper and make sure they know which agreements include Software Assurance. A good salesperson will get you an answer fairly quickly, and the experts will ask you about work-at-home initiatives, security requirements, and/or application virtualization.
- **How do I manage and track my license keys?**—This question is specifically targeted to determine whether your salesperson is aware of the tools available from Microsoft, such as the eOpen Web site and the Volume License Service Center. If they have experience working with Microsoft and solving customer issues, they will know about the tools available to the partner and user communities

- **Will I receive media with my purchase?**—Many of the Volume License Programs include a single copy of the media. Additional copies can be purchased or downloaded from the Microsoft Volume Licensing Service Center (VLSC). This may seem like a basic question with minimal value, but the key is to determine whether your salesperson is thinking in terms of both strategy and logistics. It is important to know that they are looking for ways to save you time with day-to day tasks as well as helping with strategic vision. Nothing will kill a project faster than not being able to install your software on cut-over day because no one has a media kit.

Summary

As you can see, from the wide variety of purchasing vehicles available, Microsoft has done a fairly good job at creating contracts that can aid businesses on many different levels. If you can combine the right service agreement with the right business partner, you will help realize your company's strategic vision, address challenging business issues, and deliver the best possible solution to your user community—not to mention saving yourself a lot of potentially wasted time.

Jim Varner is a technology consultant who has been involved in the industry for more than 14 years. During this tenure, Jim has discussed the design, implementation, and support of complex networks and applications with executives as well as implementation specialists. By focusing on all aspects of technology deployment, he has guided hundreds of organizations, large and small, through the pitfalls of technology deployments, to insure that any applied technology meets or exceeds set business goals. In addition to his consulting work, Jim has contributed to several publications including Redmond Magazine *and* The Northern Colorado Business Report *and has written chapters for several software installation guides. Jim holds certifications in Cisco, Citrix, EMC, VMWare, and various other products, and uses this knowledge to make business processes more efficient by applying technology solutions to specific business problems.*

Practical PowerShell

Performance Reporting with Windows PowerShell

by Jeffery Hicks

You can download a zip file with all these scripts from http://www.realtime-windowsserver.com/code/v2n2_Practical_PowerShell.zip.

In the past few months, I showed you how to use PowerShell and Windows Management Instrumentation (WMI) to monitor system events. A related task is performance monitoring and reporting. You can combine PowerShell v1.0 and a few .NET classes to get all the performance monitoring and reporting you are likely to require with only a small investment of your time.

We'll accomplish this using the System.Diagnostics.PerformanceCounterCategory and System.Diagnostics.PerformanceCounter classes. You can use these classes to monitor performance locally or for a remote system. The first step is to get a list of available performance categories on a given system. You can accomplish this step by invoking the System.Diagnostics.PerformanceCounterCategory object's Get-Categories() method:

```
PS C:\ > $categories=[System.Diagnostics.PerformanceCounterCategory]::GetCategories()
```

Use an expression such as this to discover the available performance categories.

```
PS C:\> $categories | sort CategoryName | Select CategoryName

CategoryName
------------
.NET CLR Data
.NET CLR Exceptions
.NET CLR Interop
.NET CLR Jit
.NET CLR Loading
.NET CLR LocksAndThreads
.NET CLR Memory
.NET CLR Networking
.NET CLR Remoting
.NET CLR Security
.NET Data Provider for Oracle
.NET Data Provider for SqlServer
```

```
Battery Status
BITS Net Utilization
Browser
Cache
Database
Database ==> Instances
Database ==> TableClasses
Distributed Transaction Coordinator
Generic IKE and AuthIP
HTTP Service
HTTP Service Request Queues
HTTP Service Url Groups
ICMP
ICMPv6
…
```

I've truncated the output and your results will likely vary from mine. Each of these categories will have a collection of counters. Let's see what counters are available for the System category.

```
PS C:\> $category=New-Object System.Diagnostics.PerformanceCounterCategory "system"
PS C:\> $category.GetCounters()
```

You'll see quite a list of counter information. Or you might prefer a simpler display of information:

```
PS C:\> $category.GetCounters() | select CounterName,CounterHelp
```

Before you get too excited, there is one little gotcha here. You can only retrieve the counters from a given instance of a category. Some categories, like System, are single instance. Obviously, you can only have one system at a time. But other categories such as Process are multi-instance. If you run the GetCategories() method again, you'll see a property called CategoryType. This is your clue to discovering whether a category is single instance. If it is a multi-instance category, you will need to get an instance for the GetCounters() method. First, let's create a PerformanceCounterCategory object for Process.

```
PS C:\> $category=New-Object System.Diagnostics.PerformanceCounterCategory "process"
```

Next, we'll get a list of all instances of this category:

```
PS C:\> $category.GetInstanceNames()
MSASCui
explorer
audiodg
ehmsas
taskeng
System
winscp
```

```
Communications_Helper
VCDDaemon
vmware-remotemks
…
```

We could use a specific instance to get the counter names, but instead we'll simply take the first instance in the list.

```
PS C:\> $category.getcounters($category.getinstancenames()[0]) | Format-Table
CounterName,CounterHelp -wrap -autosize

CounterName             CounterHelp
-----------             -----------
% Processor Time        % Processor Time is the percentage of
                        elapsed time that all of process threads
                        used the process or to execution
                        instructions. An instruction is the basic
                        unit of execution in a computer, a thread
                        is the object that executes instructions,
                        and a process is the object created when
                        a program is run. Code executed to handle
                        some hardware interrupts and trap
                        conditions are included in this count.
% User Time             % User Time is the percentage of elapsed
                        time that the process threads spent
                        executing code in user mode.Applications,
                        environment subsystems, and integral
                        subsystems execute in user mode. Code
                        executing in user mode cannot damage the
                        integrity of the Windows executive,
                        kernel, and device drivers. Unlike some
                        early operating systems, Windows uses
                        process boundaries for subsystem
                        protection in addition to the traditional
                        protection of user and privileged modes.
                        Some work done by Windows on behalf of
                        the application might appear in other
                        subsystem processes in addition to the
                        privileged time in the process.
…
```

To simplify these steps, I've created scripts, which you can download from the Real Time Web site. The first is a function to display performance category names and help descriptions.

```
#Show-PerfCategories.ps1

Function Show-PerfCategories {
  Param([string]$computername=$env:computername)
```

```
$perfCategories=[system.diagnostics.performancecountercategory]::GetCategories($computer
name)

$perfCategories | Sort CategoryName | select CategoryName,CategoryHelp

}
```

Load the function into your PowerShell session and run:

```
PS C:\show-perfcategories
```

You should get a list of performance categories. The function defaults to the local computer, but you can specify a remote computer:

```
PS C:\show-perfcategories FILE23
```

Once you've discovered the categories, use List-PerfCounters.ps1 to display counters for a specific category.

```
#List-PerfCounters.ps1

#usage
#list-perfcounters.ps1 -category "memory"
#list-perfcounters.ps1 -category logicaldisk -computername "FILE01"
#list-perfcounters.ps1 -category logicaldisk -computername "FILE01" |
# format-table CounterName,CounterHelp -autosize -wrap

Param([string]$category="System",
    [string]$computername=$env:computername
    )

$perfcategory = New-Object System.Diagnostics.PerformanceCounterCategory($category,$compu
ternname)

if ($perfcategory.categorytype -match "SingleInstance") {
    $counters=$perfcategory.getcounters()
}
else {
#get first instance if none specified
    $instance=$perfcategory.getinstancenames()[0]
    $counters=$perfcategory.getcounters($instance)
}

write $counters
```

Finally, we have all the pieces. Now we need to get data. We will need an instance of the System.Diagnostics.PerformanceCounter class.

```
PS C:\> $counter=new-object system.diagnostics.performancecounter “process”,”Thread
Count”,”powershell”,”CHAOS”
```

This command creates a PerformanceCounter object for the Process category. The counter is “thread count” and I’m getting the information from the computer named CHAOS. The computername is optional. However, because the Process category is multi-instance, I need to specify an instance, in this case, the PowerShell process.

Performance counters typically can return information in a raw format. You can get this data by calling the NextSample() method.

```
PS C:\> $counter.nextsample()


RawValue         : 11
BaseValue        : 0
SystemFrequency  : 14318180
CounterFrequency : 10000000
```

```
CounterTimeStamp : 128739185817018577
TimeStamp        : 422622118184
TimeStamp100nSec : 128739185817018577
CounterType      : NumberOfItems32
```

However, I think you'll find the "cooked" format easier. The NextValue() method retrieves the next performance sample and formats it into a more user-friendly format.

```
PS C:\> $counter.nextvalue()
11
```

So that you don't have to manually run through the steps I've been demonstrating, I wrote the following script, Get-PerfCounterData.ps1. This script will return all performance counters and values for a given category on a given computer.

```
#Get-PerfounterData.ps1

Param ([string]$category="System",
       [string]$instance,
       [string]$computername=$env:computername
       )

Function New-PerfDataObject {
    Param([string]$computername,
          [string]$category,
          [string]$instance,
          [array]$counters
          )

    #create empty object
    $obj = New-Object PSobject
    $obj | Add-Member NoteProperty -name "Computername" -value $computername.ToUpper()
    $obj | Add-Member NoteProperty -name "Category" -value $category.toUpper()
    $obj | Add-Member NoteProperty -name "Instance" -value $instance

    #get counters and data
    foreach ($counter in $counters) {
        $value=(New-Object System.Diagnostics.PerformanceCounter $category,`
        $counter.countername,$instance,$computername).NextValue()
        #create a property for each counter and add it to the object
        $obj | Add-Member NoteProperty -name $counter.countername -value $value
        }

    #write the performance data object to the pipeline
    write $obj

}
```

```
Trap {
    #continue and ignore errors
    Continue
}

$perfcategory = New-Object System.Diagnostics.PerformanceCounterCategory($category,$compu
ternname)

#if category doesn't have a category type it likely doesn't exist, display a warning and
exit.
if (! $perfcategory.categoryType) {
    Write-Warning ("{0} is not a valid performance counter category on {1}" -f $category.
ToUpper(),$computername.ToUpper())
    $categories=[system.diagnostics.performancecountercategory]::GetCategories($computern
ame) |
     sort CategoryName | select CategoryName
    [string]$categorystring=""
    $categories | foreach {$categorystring+=$_.categoryname + ", "}
    #remove trailing comma
    $categorystring=$categorystring.Remove($categorystring.LastIndexOf(","))
    Write-Warning ("Valid categories are {0}" -f $categorystring )
    Return
}

#if single instance, get peformance counters
if ($perfcategory.categorytype -match "SingleInstance") {
    $counters=$perfcategory.getcounters()

}
else {

    #get first instance if none specified so we can get the counters
    if (! $instance) {
        $instances=$perfcategory.getinstancenames()
        $counters=$perfcategory.getcounters($instance[0])
    }
    else {
    #otherwise get counters using specified instance
        $counters=$perfcategory.getcounters($instance)
    }

}
```

```
#enumerate performance data for all instances of the given category
if ($instance) {

    New-PerfDataObject -computername $computername -category $category `
    -instance $instance -counters $counters

}
else {

    foreach ($instance in $instances) {
    #create a performance data object for each instance

     New-PerfDataObject -computername $computername -category $category `
    -instance $instance -counters $counters

    }

}
```

The script uses the techniques we've been examining and creates a custom object. The object's properties are the counter names. If the category has multiple instances, a performance data object is written to the pipeline for each one. The script defaults to the System category for the localhost.

```
PS C:\> c:\scripts\posh\get-perfcounterdata


Computername                : CHAOS
Category                    : SYSTEM
Instance                    :
File Read Operations/sec    : 0
File Write Operations/sec   : 0
File Control Operations/sec : 0
File Read Bytes/sec         : 0
File Write Bytes/sec        : 0
File Control Bytes/sec      : 0
Context Switches/sec        : 0
System Calls/sec            : 0
File Data Operations/sec    : 0
System Up Time              : 0
Processor Queue Length      : 0
Processes                   : 98
Threads                     : 1113
Alignment Fixups/sec        : 0
Exception Dispatches/sec    : 0
Floating Emulations/sec     : 0
% Registry Quota In Use     : 7.089407
```

But you can specify any category (remember to put any categories with spaces in quotes) and any computer.

> Note: You've probably noticed that unlike WMI, I haven't offered any techniques for specifying alternate credentials. All of my code assumes you are running PowerShell with credentials that have administrative privileges on any remote machine. There are ways to authenticate with the .NET Framework, but it takes more coding and I didn't want to add any more complexity than was necessary for an already complex topic.

One easy alternative is to map a network drive in Windows using alternate credentials. If you have a secure connection to another server, PowerShell and the scripts I've shown you will use the connection's credentials.

Here are some samples you might want to try. Substitute an appropriate computer name when necessary. Remember that some categories are OS- and application-specific, so I'm not going to guarantee you'll get results for all these commands.

```
PS C:\Scripts\PoSH> .\get-perfcounterdata "logicaldisk" -instance "C:"
PS C:\Scripts\PoSH> .\get-perfcounterdata -category "tcpv4" -computername "Vista02"
PS C:\Scripts\PoSH> .\get-perfcounterdata -category "logicaldisk" -computer "FILE23"
-instance "F:"
PS C:\Scripts\PoSH> .\get-perfcounterdata -category "paging file" -instance "_total"
```

I intentionally developed the script so that it could write to the pipeline. This allows you to pipe the script's output to other PowerShell cmdlets.

```
PS C:\Scripts\PoSH> .\get-perfcounterdata -category "process" -computer "XP01" | export-
clixml c:\test\XP01procperf.xm
```

This will get performance data for all processes on XP01 and export to an XML file. Or how about this:

```
PS C:\Scripts\PoSH> .\get-perfcounterdata -category "logicaldisk" -computer "FILE02" |
Sort Instance| format-table Instance,*Free* -autosize

Instance % Free Space Free Megabytes
-------- ------------ --------------
_Total       9.400517          32281
C:           41.48713          15829
F:           5.389822          16452
```

We can also put the script in the middle of the pipeline.

```
PS C:\Scripts\PoSH> get-content servers.txt | foreach {.\get-perfcounterdata -category
"processor" -instance "_Total" -computername $_} | convertTo-HTML -title "Processor
Performance" | out-file c:\reports\procperf.htm
```

This expression will get process performance for every computer in servers.txt. The output will be converted to HTML and written to a file.

If you have PowerGadgets, you could also send output to a chart or dial and create your own small operations center. Although you can use Get-PerfCounterData as written, I encourage you to use it as a learning or prototyping tool for additional performance monitoring scripts.

If you need help with any of these scripts or commands, or assistance in developing your own performance monitoring and reporting scripts, you are welcome to use the PowerShell forum at ScriptingAnswers.com.

Jeffery Hicks, MCSE, MCSA, MCT, and Microsoft PowerShell MVP, is a Scripting Guru for SAPIEN Technologies. Jeff is a 16-year IT veteran. He has co-authored and authored several books, courseware, and training videos on administrative scripting and automation. His latest book is WSH and VBScript Core: TFM (SAPIEN Press 2007). You can contact him at jhicks@sapien.com.

Exclusively Exchange

Designing for Disaster Recovery

by J. Peter Bruzzese

With all the buzz around high-availability for Exchange 2007, some have wondered whether a discussion of disaster recovery is even necessary and if backups are obsolete—ancient history for Exchange. It is true that you can utilize the high-availability options in Exchange 2007, such as Cluster Continuous Replication (CCR), combined with Standby Continuous Replication (SCR) to provide such a solid high-availability solution that you could almost say you are ready for a disaster.

However, there are many reasons that setup might not be possible for your particular situation. Your environment might not truly need several Exchange servers (which you would need in order to establish CCR) nor a reason for a secondary site and location (which you would want if you were going to make full use of SCR) nor the budget for the hardware and software that is required to implement such solutions (considering multiple servers and the fact that you need to run Enterprise Edition of Windows Server to implement the CCR cluster). Disaster recovery solutions may be what your environment should design. But what does that entail?

Assuming that the three primary recovery points are messages, mailboxes, and/or databases (within storage groups), let's consider some of the options you have at your disposal for recovering from a disaster. Keep in mind that a disaster might be of the natural or man-made sort or an accidental occurrence such as a database corruption, a disk failure, or a server failure.

Disaster Recovery Solutions

Typically, the primary server to be concerned about amongst your server roles is the Mailbox Server, so that will be our focus. However, there are non-Mailbox Server roles that you would also want to consider for restoration in the event of a disaster. For the purposes of this discussion, we will consider the following items:

- Backup/recovery solutions
- VSS backups
- Database portability
- Recovery storage groups
- Dial-tone recovery

Backup/Recovery Solutions

In terms of backing up your Exchange databases and mailboxes, with legacy Exchange implementations (Exchange 5.5, 2000, and 2003), your built-in backup/recovery tools within the Windows Server OS (NTBackup) was enhanced to allow for the backup of your Exchange storage groups. Such continues to be the case if you have installed Exchange 2007 on Windows Server 2003. You can perform a streaming backup (not a VSS backup) of the local system. However, Server 2008 does not (at the time of this writing) support a streaming online backup of the Exchange databases. Thus, you will have to pursue a third-party solution.

Note: The Microsoft Exchange team has mentioned that a future fix will correct this shortcoming. While you wait, if you want to use Windows Server 2008, you will need to consider third-party alternatives for backup. One consideration is Microsoft's System Center Data Protection Manager 2007.

VSS Backups

Volume Snapshot Service (typically called Volume Shadow Copies) is a solution that is part of the Windows OS; it is used already to provide a modicum of recoverability in terms of system restore and/or previous versions; however, in relation to Exchange, the functionality is built-in to be taken advantage of by Server 2003 and 2008. Be aware that this requires a third-party solution to utilize the VSS ability of taking a point-in-time snapshot of your data.

Note: There are a variety of third-party backup solutions for Exchange but not all take advantage of VSS functionality. One example is Robobak, which provides a thin-backup solution for Exchange that is agentless, designed for remote and branch office data protection, offers "incremental forever" backups, provides de-duplication for single instance of data, and has on-demand synthetic restores. This option is an excellent solution, but one that doesn't currently use VSS, so be aware of your needs and wants and look for a solution that fits.

Database Portability

Before discussing the virtues of recovery storage groups, it is good to mention here that Exchange 2007 has simplified database portability to such a tremendous degree that you can actually restore any mailbox database to any server within the same organization. Although previous versions required you to have a server with the same name, such is not the case with Exchange 2007. If you need to relocate a database due to maintenance or disaster, you can move it from one storage group to another, from one disk to another, or from one server to another.

Recovery Storage Groups

Rather than restoring a storage group directly to a server, there are times when you might need only a mailbox or message to be restored. In these situations, you don't have to restore your backup to a dummy server; you can restore it to a recovery storage group and then move the message, mailbox, or database (mailbox not Public Folder) over to the functioning database.

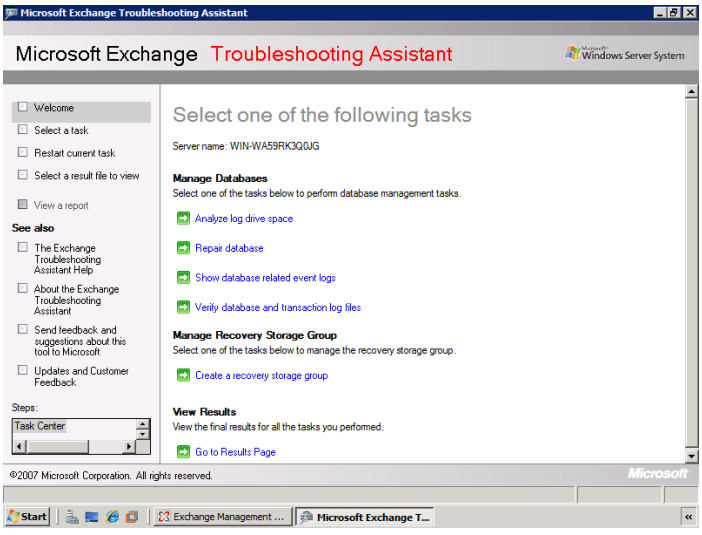

Figure 1: Creating a recovery storage group.

To accomplish this task, you need to work through your Exchange Consoles Toolbox from the Database Recovery Management Tool. You need to first create the recovery storage group (as you see in Figure 1), and then you can restore the backup to that storage group.

Dial-Tone Recovery

If your mailbox servers crashes, you need to get your users up and running fast with the ability to send and receive email. If that is your primary concern, you can perform a dial-tone recovery. Because the actual mailbox data is part of Active Directory (AD), you can restore the mailbox configuration information very quickly and then restore the mailboxes at leisure (well, not really leisure, but without users breathing down your neck about sending and receiving email). Keep in mind with the dial-tone recovery that users will open Outlook and will see no mail from the past (because that hasn't been restored). You should warn them of this fact. Later on when the database is restored, you are going to swap the new and old databases. However, you are still going to need to recover the day's mail and put it in the newly restored databases. This task involves the same process as restoring mail from a standard recovery. So, you restore the database to the recovery storage group, you swap the restored older version with the newer day's mail version, and then you merge over the day's mail with the recovered database. Sounds a bit complicated but it's just a matter of following the proper steps.

Don't Panic…Plan Ahead

The key to disaster recovery is to know your options and then set about planning for the worst. Planning involves ensuring you have everything documented (and the documentation is held offsite), everything backed up or duplicated in one way or another for recovery (and again, offsite, and if you are really worried, out-of-state or country), and you run practice drills to ensure you know exactly what to do and how.

J. Peter Bruzzese is an MCSE (NT, 2K, 2K3)/MCT, and MCITP: Enterprise Messaging Administrator. His expertise is in messaging through Exchange and Outlook. J.P.B. is the Series Instructor for Exchange 2007 for CBT Nuggets. In harmony with the joy of writing Exclusively Exchange for Realtime Publishers, he has created a free Exchange training site at www.exclusivelyexchange.com. He is co-founder of ClipTraining.com, a provider of short, educational screencasts on Exchange, Windows Server, Vista, Office 2007 and more. You can reach Peter at jpb@cliptraining.com.

ExclusivelyExchange.com Free Training Videos

Would you like to learn more about ActiveSync? Check out these free training videos at www.exclusivelyexchange.com: "Planning for Disaster," "Recover Deleted Items from the Outlook Client," "Backup and Recover Data (Server 2003 and 2008)," "What Is VSS?," "What Is Database Portability?," "What Is a Recovery Storage Group?," "Create a Recovery Storage Group," "Working with Recovery Storage Group Options," "What Is a Dial-Tone Recovery?," and "Restoring a Non-Mailbox Role."

Copyright Statement