Realtime publishers

January 2009
Volume 2, Number 1

Windows Administration
in Realtime

When people find something good,

word gets out.

vmware®

VMware makes Windows better.

Over 100,000 companies of all sizes, including 94% of the Fortune 1000, use VMware. Why? Because VMware® Infrastructure delivers greater availability, superior manageability and the broadest support for legacy and current Windows applications.

And when you deploy VMware, you get a proven brand that's the global leader in virtualization—and has been for over a decade. It's experience that matters.

Learn more about the benefits of virtualizing your Windows environment with VMware.

Get your free "getting started kit".
Find out more at www.vmware.com/go/winsmb

Redmond MAGAZINE
Best of the Best
READERS' CHOICE Awards
Winner
2008

Letter from the Editor

The Plight of the Misunderstood IT Pro

by Greg Shields

Have you ever come home from a hard day's work, sat down at the dinner table, and started talking to your wife/kids/partner/significant-other, only to have their eyes glaze over at the first talk of technology. Ever accomplished something spectacular in your work life that none of your friends ever truly appreciate?

That's the plight of the misunderstood IT pro.

I hear your pain. I've been working in the trenches of IT for some 15-odd years now, and I still catch myself devolving into techno-speak on occasion with friends at the bar. Ever done this fun little trick? Start down the road of a super-techie IT story, only to watch your friends melt before your mastery of useless information. I have, actually enough that these days the ol' friends catch it early and quickly move on.

But, while humorous, the misunderstood IT pro is the lifeblood of our businesses and our society. Without us, business these days couldn't run, and friends' and family computers would never get fixed after "accidentally" finding themselves on one of those quasi-legitimate Web sites nobody ever admits to.

It's a new year here at the *Windows Administration in Realtime* eJournal, and in honor of it, we salute you the IT professional. We know that without you, the world would stop on its axis, and Google itself would crash and burn in a pile of its own heat generation. Remember that even though you, the IT professional, might be misunderstood from time to time, like it or not, you'll always be valued.

Answers from the Experts

Can changes to Active Directory go through an approval process?

by Don Jones

Q: Is there a way to prevent changes from being made to Active Directory until those changes have been reviewed and approved?

A: Not natively, no. In Active Directory (AD), any Domain Admin or Enterprise Admin, or anyone with the proper delegated permissions, can make any changes without the opportunity for review or approval. That makes AD a bit tough to fit into corporate management processes that have a review and/or approval phase; or rather, AD can "fit" into these processes, but the requirement for someone to wait for a review or approval is essentially voluntary.

Commercial tools exist to add this functionality to AD. Microsoft's upcoming Identity Lifecycle Manager "2," for example, includes workflow capabilities, as does the ActiveRoles Server product from Quest Software. Of course, these products add a lot more than just change control, which is what you're asking about.

If you just need change control, look at the ADMC, a free tool from TurboChargeAD.org. It consists of a server-side component, which is ideally installed on your domain controllers, and a client-side console. It also extends the Active Directory Users and Computers (ADUC) console. ADMC provides two primary pieces of functionality: change control and automated business rules.

With the change control element, you designate which operations trigger a change control workflow,

such as creating a new user. You also designate the groups that are allowed to perform the triggering operation, meaning you can select groups other than those that would normally be able to create a user (or whatever). You're technically not giving them permission in AD to perform that action; you're instead giving them permission within ADMC to create a request to perform that action. From there, you set up a review and/or approval phase. Within this phase, you can specify the number of reviewers/approvers needed to advance the request, specify notification options for requests that are advanced or rejected, and so forth. Finally, you specify whether approved changes are committed automatically or queued up and committed on a schedule.

With the business rules element, you set up what look like Outlook mail filters. Instead of moving mail to a folder, though, these rules might auto-populate AD attributes when a new object is created, halt the creation of new objects that don't have specific attributes, halt the deletion of organizational units (OUs), or whatever you want. The rules can be as basic or as complex as you want, and they're carried out automatically on each domain controller.

With the ADMC's server-side component deployed to each domain controller in your environment, it becomes pretty much impossible to bypass either the workflow or business rules. Such is the case even if someone's using a script (rather than ADMC or ADUC), command-line utility, or anything else to modify the directory.

It's unfortunate that AD doesn't include native capabilities such as these; I think this type of workflow and rules processing is the minimum level of functionality anyone really needs to manage AD in today's process-oriented environments. But at least ADMC adds that minimum functionality at no cost.

Do you have an IT question you'd like Don to answer? Send it to answers@realtimepublishers.com for consideration!

Don Jones is a co-founder of Concentrated Technology (www.concentratedtech.com), helping to deliver IT knowledge in less time using innovative content techniques. He also serves as CTO and Series Editor for Realtime Publishers. Don is the author of more than 30 IT books, including Windows PowerShell: TFM; VBScript, WMI, and ADSI Unleashed; Managing Windows with VBScript and WMI; and many more. He is a multiple-year recipient of Microsoft's "Most Valuable Professional" (MVP) Award with a specialization in Windows PowerShell.

Product Review

PowerShellPlus Professional Edition, Version 2

by Hal Rottenberg

When it comes to systems administration and automation, one of the most important tools in an admin's belt is the Integrated Scripting Environment (ISE). This term is being popularized by Microsoft with the coming release of version two of Windows PowerShell. Generally speaking, an ISE is the sibling to an Integrated Development Environment (IDE). Many of you will recognize an IDE as a comprehensive software package that helps developers program software. Some of the most popular examples are Microsoft Visual Studio and the Java-based Eclipse IDE. An ISE, however, is made for systems administrators: The focus of the application will be more on performing actions and simplifying tasks, and less on editing screens of text or debugging lengthy programs.

PowerShellPlus from Idera is an excellent example of the ISE software genre. Let's walk through some of its features.

Documentation You Will Actually Use

When you start PowerShellPlus, you are immediately greeted with the PowerShellPlus Learning Center (see Figure 1: Learning Center). At first glance, this is your basic built-in help system with a table of contents and search functionality—all of which is standard for Windows applications. However, PowerShellPlus has brought much more to the table. In addition to the "Getting Started" section and a user's guide, there are several pages of original PowerShell tutorials and examples.

If you are already familiar with PowerShell, you know that it has a very comprehensive help system, which you can access via the console. The Learning Center actually grabs that content and it is presented alongside the rest of the help topics. This applies to the conceptual documentation provided by Microsoft as well as the help files for every cmdlet—including cmdlets from third-party "snap-ins."

In addition, you will find detailed WMI documentation including descriptions of the properties and methods of hundreds and hundreds of WMI classes. If you know the name of one, such as "Win32_Process," you can type that into the search box to quickly find what you need.

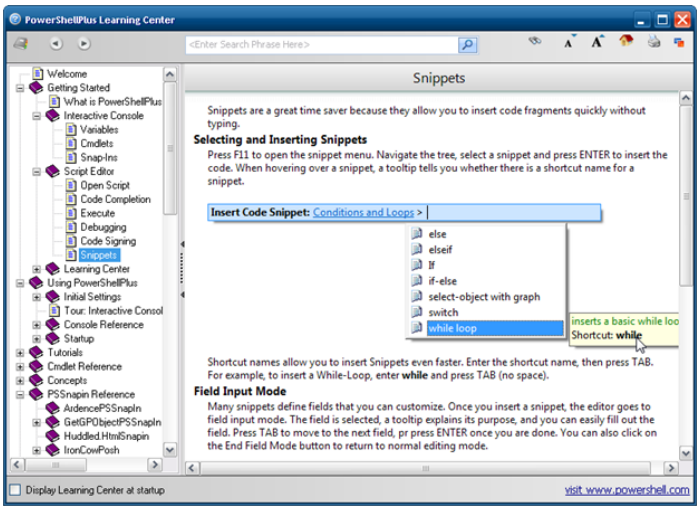

Figure 1: Learning Center.

It's All About the Console

Once you close the Learning Center, you are presented with a console window—plus several new toys that I'll go over in a moment. It is obvious right away that a primary design goal of this software is to really push the envelope when it comes to the concept of a shell. And as you will see, each of these pieces comes together to form a very nice package.

As you might have guessed based on the name, PowerShellPlus is designed specifically for use with Windows PowerShell. Although PowerShell is the best thing to happen to Windows IT pros since the GUI, the PowerShell console and the trusty CMD shell (both of which are based on the same console subsystem), are ugly and featureless. Even Windows Vista with its new eye candy did not change much in this regard.

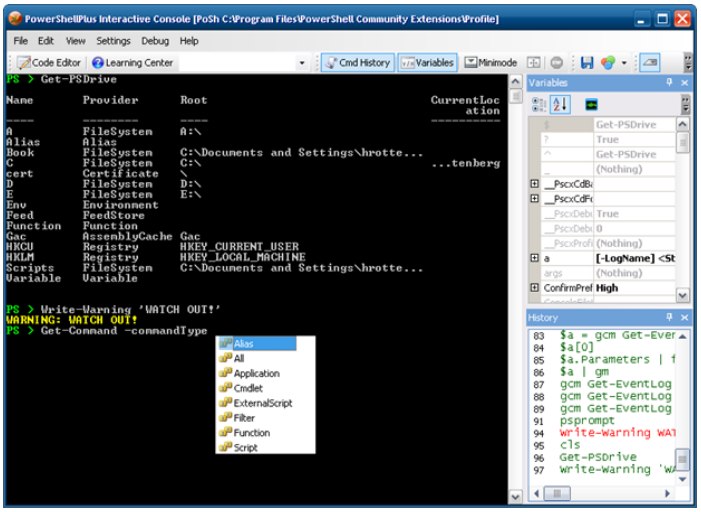

Figure 2: PowerShellPlus Console.

As you can see in Figure 2: PowerShellPlus Console, you have a conventional shell that is surrounded by a menu strip and toolbar on the top and two panes on the right titled Variables and History. You can liken this window to your basic shell, but the extras are quite obvious. The first thing in this screenshot that might jump out at you is the box in the middle of the console with several lines of text. If you look at the example here (the Get-Command cmdlet), you'll see that this box actually contains a list of the acceptable values to the "commandType" parameter. This is just one aspect of an intelligent code completion feature that works similarly to Microsoft's IntelliSense feature in Visual Studio. You will find that the code completion knows the names of every PowerShell cmdlet in your session, as well as their parameters, and in some cases, the permitted values to those parameters as shown in the figure. In addition, you can use the tab key to invoke completion of filenames, variables, and more. All of these items are automatically generated on the fly. So, for example, if you add a snap-in to your session, PowerShellPlus will detect the new cmdlets and you will be able to use the code completion with them right away.

The next features that deserve mention in the console are the two panes on the right side of the screen. If you are the minimalist sort, you can certainly get rid of them. But I would advise that you spend a little bit of time getting used to them first. Note that you can move these panes, detach them, or use the pushpin icon to have them slide into drawers on window's edge.

The Variables pane holds a list of the variables in your current PowerShell session. It's a dynamic list, and the contents are updated every time you enter a command. There is also a very cool button that will change the mode to display the current contents of the PowerShell pipeline. If you click that button, and then type "dir" into the shell, the Variables pane will list every item in your current working directory. But wait, there's more! Each item in the list represents a PowerShell object, and you are able to browse the properties of those objects (such as filename or length), by clicking the expand button.

Below that, you will find the History pane. This box holds a list of every recent command that you have typed into the console, up to the maximum, which is specified in a special PowerShell variable called $MaximumHistoryCount. What's more, this list persists after you close and re-open the program. This is a great way to retain the context of something you were working on previously. As with the PowerShell console, these previously executed commands can be accessed with standard up/down cursor controls. In addition, you can drag commands from the history pane into the console to perform a paste. PowerShellPlus goes even further with a context menu that allows you to copy multiple commands and send them to the built-in script editor.

It Slices, It Dices, It Edits Scripts!

The PowerShellPlus script editor is a separate window that can be shown or hidden with the press of a toolbar button or the Ctrl+E hotkey. This novel approach gives you an "always ready" capability to edit scripts without getting in the way when you need to execute commands in the shell.

It's not until you use the console window in combination with the built-in script editor that you really see PowerShellPlus shine. The ability to test a line of script in an interactive console, and then quickly transfer it to the editor and expand upon it, is almost priceless. In fact, you can go the other way too. Figure 3: The script editor shows how you can optionally embed the console window inside of the script editor. This functionality allows you to quickly see the results of the code you are composing without worrying about overlapping windows. Also note here how the WMI help is shown on-screen during code completion.

The editor has many of the features that one would expect from a modern text editor, such as code folding and colorized syntax highlighting (for PS1, XML, and a few other file types). Of course, you also have code-completion, just like in the shell. In addition, you get an important feature that is normally only found in an IDE—a debugger.

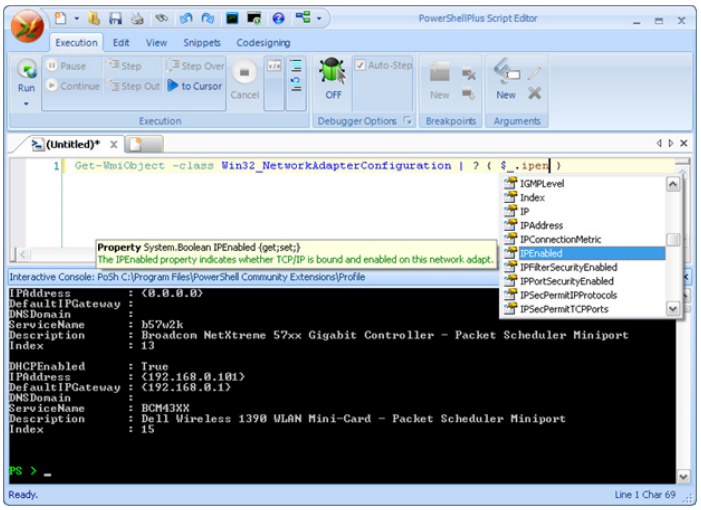

Figure 3: The script editor.

Debugging code is a concept that is alien to most systems administrators, so this is an area where you will see divergence between an IDE and an ISE. The debugger in PowerShellPlus strikes a good balance between ease-of-use and functionality. Administrators will find it very useful when trying to unravel some of their longer scripts.

Conclusion

PowerShellPlus has a lot to offer to IT pros of all levels. The only bugs that I experienced were related to using pre-release software. (What can I say—I'm a glutton for punishment.) Performance overall is quite speedy, although when my workstation starts to swap memory to disk, the program occasionally takes more than 10 seconds to recover its senses and respond to keyboard and mouse clicks. However, it's hard to say for certain that this is the fault of the application and not the operating system (OS) or other software running in the background.

At $174, PowerShellPlus Professional Edition is priced beyond the budgets of all but the most dedicated of hobbyists. But when viewed in the context of the time-saving features it brings as well as the fact that a combination command shell and text editor is something admins will use for hours every day, the price seems to be justifiable for most business scenarios.

Disclaimer: As of this writing, Idera is a paid sponsor of the PowerScripting Podcast, of which this author is a co-host. However, I have been a beta-tester and power user of PowerShellPlus since its earliest days in development, long before Idera was involved in marketing of the software.

Hal Rottenberg is an Admin Frameworks MVP living in Woodstock, Georgia. He is well-known in sysadmin circles for co-hosting the PowerScripting Podcast and heading up the PowerShellCommunity.org website. He is also currently writing a book titled "Managing VMware Infrastructure with PowerShell: TFM " to be published by SAPIEN Press.

Product Review

NetCmdlets

by Marco Shaw

When Windows PowerShell v1 was officially released by Microsoft in November 2006, /n software was one of the first companies ready at launch time with their own third-party extensions to PowerShell, named NetCmdlets.

Based on /n software's rock-solid and popular IP*Works! product, the first official release of NetCmdlets was version 1, and it includes more than 30 cmdlets. It is currently their stable release. What sets NetCmdlets apart is that most of their cmdlets are networking related—something definitely missing in PowerShell for most automation tasks. The following list highlights some of the features that v1 provides:

- SNMP device management
- Remote Secure Shell access
- LDAP directory access (for OpenLDAP servers and Active Directory)
- Email send and receive (using IMAP, POP, and SMTP)
- FTP file transfer
- Instant messaging
- Network monitoring
- Web services through HTTP and RSS
- DNS configuration
- Encoding and decoding capabilities (including MIME, UUEncoding, URL, Hex, etc.)
- File compression (including password protection, AES Encryption, and 4GB+ archive support)

Included with v1 is a help file in CHM format and demo PowerShell scripts to help users to begin using their cmdlets. They provide their own customized PowerShell console with NetCmdlets already loaded, but loading NetCmdlets in a default PowerShell console is as easy as running:

```
PS>Add-PSSnapin NetCmdlets
```

You can also add this command directly into your PowerShell profile so that the cmdlets are loaded automatically every time you start a PowerShell console.

Shortly after the release of NetCmdlets v1, /n software released a PowerShell remoting solution, which is definitely useful for organizations wanting to use PowerShell to manage large environments. The remoting product uses the SSH protocol for increased security. One very interesting feature of their remoting solution is that most SSH clients are also supported. For example, imagine using a BlackBerry with a third-party SSH software application to remotely connect to a Windows server, and be able to interact with PowerShell on that remote system for remote administration. I have already tested it and it works!

Since releasing their remoting product, a beta of NetCmdlets version 2 was released. Along with improvements to the existing v1 cmdlets and the addition of some new cmdlets, the PowerShell remoting solution was rolled into the NetCmdlets product. More specifically, the v2 beta includes cmdlets for dealing with Amazon S3 online storage and cmdlets for interacting with their SSH remoting solution. Improvements were also made to several of the v1 cmdlets to support pipeline input, which adds a lot of flexibility when doing PowerShell scripting.

For the more advanced PowerShell user, the NetCmdlets cmdlets output rich objects that can participate in PowerShell pipelines. As a result, NetCmdlets will work with all the core PowerShell cmdlets, and should work with other third-party PowerShell add-ons, depending on what type of objects these other cmdlets expect. In addition, NetCmdlets follows the Microsoft best practices for cmdlet naming and built-in PowerShell help. For example, I'm going to discuss the get-nntp cmdlet shortly, and accessing the built-in help and examples is as simple as doing get-help get-nntp from within a PowerShell session.

An interesting usage example came up several months ago when I wanted to try to gather statistics from an online discussion group—more specifically an NNTP-based discussion group. I noticed that NetCmdlets had a get-nntp cmdlet that could work with this particular protocol. Figure 1 demonstrates how I used NetCmdlets to retrieve the latest 500 NNTP postings, then I grouped the posting by the day of the week (I could have used other methods to accomplish this), and finally, I displayed the day of the week, the number of postings for that day, and a quick and dirty distribution chart to add a visual effect to the output. All of this in one single line of PowerShell code!

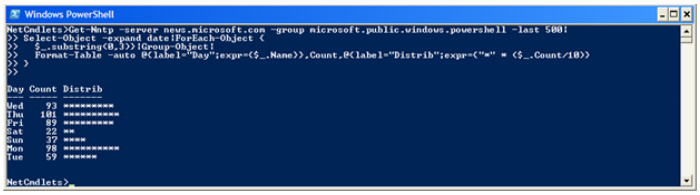

Figure 1: Using NetCmdlets.

The /n software Web site has an excellent 13-minute video that shows examples of using NetCmdlets. The Web site also has online examples that complement the packaged demo scripts.

/n software is offering free licenses for non-commercial use, so there's no excuse not to go and check out this product. For commercial use, a workstation license is $99. For server licenses, the prices are $299 for a single CPU server, and $599 and $999 for 2 and 4 CPU servers, respectively. /n software offers free email support along with paid premium support contracts (see the Web site for more details). For more information, go to the /n software Web site where a more complete list of the included cmdlets can be found.

For more examples, check out http://www.lanceroBinson.net/Tags/NetCmdlets/default.aspx and http://marcoshaw.blogspot.com/search/label/netcmdlets.

Marco Shaw is a Microsoft MVP, and Co-Director of the PowerShell Community Web site. His main blog is at http://marcoshaw.blogspot.com.

Notes from the Field

Building an Enterprise SCCM Hierarchy

by Eric Schmidt

Before I begin discussing the rollout of Microsoft Systems Center Configuration Manager 2007 (SCCM) in my organization, I first want to provide a little background. The organization I work for has multiple business units, each operating independent of the others. In many respects, these units can be viewed as individual companies with their own culture, processes, and standards. In the context of SCCM, some divisions had deployed Microsoft's previous solution, SMS, and were planning an upgrade, while others were using other systems management products.

The challenge with our project was to upgrade existing SMS environments to SCCM and deploy SCCM to those divisions that were using other products. The final goal was to unify all SCCM sites under an enterprise central site. Although this task appeared straightforward, the project proved interesting.

So where did we begin? Each division had different processes and standards for managing their environments, so the first thing the team had to do was collect data from each division for all aspects that would be included in the new enterprise model. Once this task had been completed, we were then able to identify the things that were common and focus on those that weren't. This was a critical first step because before a single server could be upgraded or deployed, we had to define and agree on the standards that would be used in the new model.

The second critical step in the design phase was to define the new hierarchy. This process begins by documenting the existing hierarchy(s) and identifying areas where consolidation can take place. In my company, we have hundreds of physical locations of varying sizes and levels of connectivity. Sites with good connectivity but a small number of clients would only get distribution points, but if they had a large number of clients, they would get a primary site. Those with less bandwidth would then also get primary sites. In some instances, if there were a large number of clients and excellent connectivity, secondary sites would be deployed. The goal of the design was to make the most efficient use of WAN links while keeping the number of primary sites as limited as possible. To do so, we created a scale for each type of SCCM site based on how much WAN traffic was going to be created. On one end, the highest amount of WAN traffic would be clients talking directly to a remote site server, and on the other, clients would talk to a local site server, which would then forward the data over a WAN link to its parent.

Once the hierarchy was established, we then looked at the SCCM configuration itself and identified what should be standardized—everything. All aspects of SCCM need to be standardized and defined up front, including advertisement, collection, package, and query names. In an enterprise hierarchy, the most effective approach is to create all these items at that central site level. In addition, all the objects I mentioned will flow down the hierarchy, so a naming standard will organize them so that they can be easily found. An example of this with collections would look something like [Division]-[Collection Name]. All collections for a specific division would be sorted together and would be differentiated from the other divisions. The other items that would be standardized are the site settings. Although each site can have its own unique settings, consistency is crucial to success. To that end, we looked at every site setting including inventory frequency, discovery, remote control, and the level of user notification for advertisements and software updates. The final item that was passed down the hierarchy was the .MOF files. Data collected from the client passes up to the central site, so it was critical that all sites use the same .MOFs; if one site were to deviate and collect more (or less), the data on the central site would be inconsistent.

Once the team agreed upon settings, it was time to start deploying SCCM, right? Wrong. It was now time to develop a governance model and implement change control. The governance model established how the hierarchy is to be managed and by whom. The new structure was enterprise-wide, where before it was disconnected and managed by each division.

When scaling something to the enterprise, there are many benefits, but there are also new risks. As an example, a bad package or advertisement can affect the entire company where before the scope of the damage would be limited to a division. To mitigate this risk, the governance model established a board that would review, discuss, and approve everything that took place at the central site. The right to make changes or create advertisements was also limited to a small number of people.

Establishing a change control process was the next item that had to be done before a single site was deployed. The purpose of change control is to put mechanisms in place to document and approve all activities related to the hierarchy, which serves to ensure that all sites in the hierarchy are configured exactly the same and to ensure that packages and advertisements have been properly tested before deployment.

So now it's time to deploy, right? Wrong again. The next item that needed to be defined was the security model. The current environment had varying levels of security with some that were strict and others that were not. The move to an enterprise-wide hierarchy created new opportunities and risks, so there was a need to create a security model that enabled the company to leverage this new asset while at the same time protect it. To do so, we defined several roles that consisted of site administrators, creators, package creators, operators, and reporting. The rights for these roles were on scale with site administrators having all rights and the reporting roles having the fewest, and the other roles with rights in between. These roles were then applied to every level in the hierarchy with each site server having groups defined for each.

This setup enabled a waterfall security model in which central site roles had the same rights all the way down the tree. Roles that were defined at a lower level in the hierarchy would only waterfall down that particular branch. The benefit of this approach is that a small number of people have rights from the top down while others only have rights down their branch of the tree. The exception to the waterfall was the right to change site settings. Although site administrator groups were created for each site in the hierarchy, their only purpose was to facilitate the waterfall of rights for the central site administrators. The reason for this setup was consistency. We wanted to ensure that only the central site administrators had the rights to change site settings. In the old structure, this was not the case and there were many instances where changes (enabling network discovery) were made at a down-level site that were not coordinated or communicated and had negative effects on the sites above it. The new enterprise hierarchy needed to be controlled so that changes were coordinated.

The final step before deploying SCCM was to establish a hardware standard. The purpose of creating a standard was to insure that all sites in the hierarchy had sufficient resources to function. In creating the standard, we looked at the role the server would be playing (primary site, secondary site, distribution point, etc.) with the primary focus being on disk space. We had to make sure that all hardware in the hierarchy was capable of storing all the packages and collecting client data. By having a standard, the central site administrators could operate with confidence knowing that the hierarchy would support anything that was created. As part of the hardware standard, we identified how the OS, SQL, and SCCM were installed and determined the drive configuration. This would make it much easier for the central site administrator to troubleshoot problems because they know how all the servers were configured. Finally, with hierarchy and settings defined, a governance model in place, a change control process established, a security model defined, and a hardware standard determined, it was time to order hardware and start deploying SCCM.

Eric Schmidt works as Enterprise Microsoft Security Technologist, with Honors, for Raytheon Company and has worked in Information Technology for 13 years. Eric has a Masters degree in Computer Information Technology and has developed extensive experience in systems administration, engineering, and architecture specializing in Microsoft Active Directory and Systems Management. Eric has been well recognized throughout his career for his contributions to designing and implementing enterprise-wide solutions using Microsoft Windows-based technologies.

The Deep Dive

You Too Can Be a Conference Speaker!

by Don Jones and Greg Shields

If you're an IT professional, it's likely that you've attended a conference or two over the course of your career. Whether that conference was small and locally sponsored, a set of independent presentations, or a huge vendor-hosted party, all conferences are about the same in the end. You, the attendee, catch a few speakers as they carry on about IT technology. You'll likely peruse an expo floor to check out your favorite vendors' new products, and maybe score some expo schwag if you're lucky or persistent. If it's a travel conference, you might even bring your family or friends to make a short vacation out of the opportunity.

But out of all conferences, no matter the size or topic, one thing remains the same—speakers. They all have them. And there's a pretty good chance that in your hours of sitting in those uncomfortable hotel chairs, you've at least once wondered, "How do those guys get good at that job anyway?"

To be honest, even being selected for a conference speaker can be tough. Your first job is to prove to the conference chairs that you're both talented and technically qualified to talk on your subject. Once you wow the decision-makers, you've then got to actually stand up and give your presentation. If you are the type who's always wanted to stand while everyone else has to sit, we present this short guide to help you understand the challenge that lays before you, and to help you prepare to deliver the best conference sessions possible.

Understand that there's no "story" here; a lot of these "rules" are fairly disconnected from one another, and may even seem arbitrary. Rest assured that they are necessary. These are not in any particular order because that might imply that some are more important than others—they're not. Everything here is 100% required in order for you to survive your first conference presentations and be invited to speak in the future.

Still want to be a conference speaker? Read on.

Start on Time, End on Time

It's imperative that your session start promptly and run for the full length of the time allotted. Do not allow more than a minute for introducing yourself, and no more than 5 minutes at the end for Q&A. Although running significantly over is bad, running short is worse—attendees feel cheated. Rehearsing is one way to make sure your material is of the correct length—although be aware that, when you're in front of an audience, nerves will often make you pick up your pace.

Set the Stage Right

Never self-deprecate. More specifically, never let your audience feel that you're anything other than the world's leading expert on the topic you're presenting. Now you may not necessarily be the world's leading expert, but that's OK. You'll find that in many cases the "superstar" speakers you know and love aren't either. But they do know their slides, and they never tell the audience to the contrary.

Pace

In your rehearsals, develop your speaking pace. Is it 2 minutes per slide? 2.5? Whatever it is, come up with a consistent pace. If one slide is taking too long, move some bullet points off it and onto the next slide so that each slide is covered in about the same time. When you're in front of your audience, use the podium clock (or bring your own) to make sure you're sticking to your pace.

Plan

Think about how people will absorb the information you're presenting. Don't just brain dump in random order; construct a path from the basics through to more advanced material, building as you go. Anticipate common questions and work the answers into your presentation at the right time.

Rehearse

You absolutely must run through your entire session, including all demos. Do this several times until you can deliver your presentation without looking at your slides. Ensure that your demos work, and once they do, don't mess with them and risk breaking them before you deliver your session. Rehearse your demos on the same computer you'll use at the conference, in the same conditions (for example, not connected to a network, connected to an external monitor/projector running at 1024 x 768). Use PowerPoint's Presenter Mode if necessary so that you can see on your laptop what the upcoming slides are and verbally transition to them. Know that the secret to a slam-dunk presentation is all in your transitions between slides.

Stay on Track

You absolutely must avoid going off on a tangent, especially in response to a question from the audience. The temptation to do so can be strong, especially because it makes you look smarter in front of attendees, but tangents will lose most of your audience. If it's a point you'll get to later anyway, then just say so; if it's out of scope, say so. If it's something you can take up in Q&A, do.

Determine How You Do Questions

Some speakers hold questions to the end of the session. Others take them throughout the session and use them as transition points. There is no correct way. Determine before you ever start your session how you plan to take questions and how you plan to draw questions out of your attendees.

Do Not Read from Your Slides

Attendees can read. They do not need you to do so on their behalf. You need to bring something extra that isn't on your slides—expertise, analogies, metaphors, stories, and so forth all make you the expert. Along with that, slides shouldn't contain too much information: five or six bullets of one sentence apiece is enough. The slides are there to keep you on track, not to contain your entire presentation. If you can't deliver the presentation from memory, then you're not ready to present at a conference. Slides should only outline and summarize what you're saying, not replace it. Do not be tempted to put all kinds of details into your slides just so that you won't forget something important. Your numerous practice sessions will ensure that all the important details are right on the tip of your tongue.

CONCENTRATED
TECHNOLOGY

MAXIMUM KNOWLEDGE
MINIMUM TIME

MONSTERS of TECH

T E C H N O L O G Y T R A I N I N G T O U R

GREG SHIELDS

author, "Windows Server 2008: What's New | What's Changed"
TechNet Magazine columnist
Top national conference speaker

DON JONES

co-author, "Windows PowerShell: TFM"
TechNet Magazine Contributing Editor and columnist
World-renowned PowerShell speaker & trainer

Quickly learn today's most important and leading-edge technologies for Windows Administrators! Jump-start your existing Windows administration skills to **Windows Server 2008** with Greg Shields' world-famous "What's New, What's Changed" class. Start automating Windows and Active Directory administration today with Don Jones' unique, Practical Method training in **Windows PowerShell**. Go further by ramping your skills up to **Windows Server 2008 R2** and **Windows PowerShell v2**. All of this in just five days of the most unique training you'll ever enjoy!

We laugh at classes where the instructor relies on slide decks and reading from a prepared script. In our class, we don't even use a projector! Instead, you view the slides on your computer, you watch demo videos on your computer, and you work hands-on almost continuously on your computer - meaning you spend more time with your hands *on the technology* and less time craning your next to see the screen. You get lab guides, slide decks, demo videos, and virtual machines on a Passport USB hard drive - which is *yours to keep*, meaning you'll be able to take a self-paced "refresher course" anytime you want!

We're offering this training to a *maximum of 60 people* in 2009. Be one of the elite - one of the few who is truly prepared for today's latest production-class technologies. Be a MONSTER of TECH!

More info: http://ConcentratedTech.com/class

Dates announced for April • More locations & dates coming soon • Early Bird Pricing Now Available

You're On Your Own

Do not plan for your audience to interact with you by sharing stories, answering questions, or even raising their hands. Many times, they'll just stare at you. 100% of the energy in the room comes from you. If you catch an experienced speaker 10 minutes after they've finished presenting, they will probably look exhausted—there's a reason why.

This Isn't Drinks and Snacks

A presentation isn't the time for you to be casual and buddy-buddy, especially if you're new. If a stage is provided, stand on it. Although experienced, charismatic speakers can prowl around in the audience and keep the energy level up, it's extremely unusual for a newer speaker to pull it off (although everyone tries). Maintain a sense of authority throughout the session. Move around on the stage a lot, use your hands (gesturing holds the audiences' visual attention) when you speak, and so forth; don't pull out a chair and sit down. You're there to work, and work hard.

Have a Backup Plan

Demos explode. Happens all the time. Don't let it catch you with your pants down. Some speakers will use recording software such as Camtasia to record their demos in advance; that way, if the live demo bombs, they can at least throw the Camtasia up and talk through the demo with the audience, pausing as necessary to emphasize points or answer questions.

Your Demo Will Explode

Assume that your Creator is actively trying to make you look stupid by ruining your demos, and take all necessary precautions to prevent it from happening. Set up virtual machines in advance, and snapshot them to ensure they "start" in the correct state. Write down the steps of your demo, and have those steps on stage with you in case you forget. Practice, practice, practice. We preach "practice" constantly, yet only one in ten new speakers actually does it—and they're the ones that are brought back to future conferences.

Energize

Whatever you do to get your energy up—Red Bull, jumping jacks, prayer, etc.—do it before you go on stage. Keep your energy up. You're not a laid-back expert who just happens to have all the answers; you're a jazzed-up, passionate enthusiast who couldn't be stopped from doing this presentation if someone put a gun to your head. The right amount of energy will feel like too much energy, but that energy is what makes the very best speakers. Attendees will respond to whatever energy you put in the room—if it feels dead, then you probably made it that way.

Repeat Questions

When someone asks a question, no matter how loudly, restate the question. This ensures that you heard it correctly, gives you time to think about the answer, ensures that the rest of the audience heard it (you're the one with the microphone, remember), and ensures that any recording equipment will pick it up. When you're rehearsing (rehearse, rehearse, rehearse) have a spouse, friend, or coworker pop in with questions from time to time—and be sure to repeat the questions. Do this until repeating questions becomes an annoying habit that's hard to shut off.

Know When to Quit

If something (like a demo) goes wrong, give yourself 15 seconds to fix it. Don't worry, it'll feel like a lifetime. After that, move on. The audience isn't here to bask in your mistakes—they're on your side. They want information from you, and you need to get on with delivering it. Apologize once for the problem, and then forget it. Don't wallow in your grief—most attendees are happy to move on, but if you repeatedly remind them of your screw-up, they'll be happy to write it on your evals. That said, your demo blowing up isn't a big deal, because you have a backup plan, right? Right?

Care About Your Evaluations

The conference does. You should too. Evaluations are the lifeblood of conference content, making the determination of who gets invited back and who doesn't. Remind your attendees to fill out evaluations at the end of your session, and suggest that they comment on their feelings about your work. Evaluations are, after all, the best way for you to get better at presenting.

Dress Up

Dress a bit nicer than you do at work. Although really "superstar" speakers may have a particular mode of dress as part of their "act," newer speakers are usually best-served by business slacks and a long-sleeve, button-up shirt. You may think a golf shirt is perfectly dressy, but they tend to look sloppy when you're moving around in them. Audiences have a natural instinct to respect dress, so take advantage of that. If you perspire heavily, wear an undershirt and use industrial-grade antiperspirant that day, applied no more than an hour before you go on stage so that it's fresh.

Introduce

Take one minute (60 seconds!!!) at the start of your session to introduce yourself. Key points to hit are anything that makes you more qualified to be on that stage: mention past experience with whatever technology you're talking about, mention any industry awards, honors, or associations, etc. Mention books and magazine articles, or your blog if you're a regular technology blogger. Then move on. Don't ask attendees for their names—you don't need to know them.

Engender an Agenda

Agendas are nice, but don't get too obsessive over them. At the start of your session, lay out what the path looks like: "We're going to cover a few background things first to make sure we're on the same page, and then move on to this, then cover that, and finish of with those." And then move on. Don't repeat your agenda slide later—just cover the material.

Summarize

Finish off with a brief summary of the five or six key points you just made in your session. This is a nice wrap up, a chance to thank attendees, and a last chance for a few questions (you should have no more than 5 minutes left by this time). Thank them again before dismissing everyone.

Be Succinct

Telling a story or analogy? Keep it short. In fact, everything about your session should be quickly getting to the point. When you're planning your session and writing your slide deck, try to find the shortest possible path to get to the session's objectives. Think of it as making a movie adaption of a book—cut out all the side plots and secondary characters to focus on the meat: just the main characters, just the biggest explosions, just the one car chase that was really elemental to the plot. Ditch everything else.

Good Grief, Is It Really This Hard?

Well…yes. Yes, it is. Really experienced speakers do a lot of this automatically, but they do it every time, or they wouldn't be experienced they'd be back at their real jobs. Speaking is not glamorous, it's bloody hard work. It's exhausting at times, takes a great deal of time to prepare for, and requires an immense amount of effort to maintain the necessary level of expertise.

So why do experienced speakers do it? Who cares!

Ask yourself why you're doing it and you'll have your answer. Just don't think for a moment that you cannot do any of the things listed here—this is the barrier to entry, and it's why there are so few professional speakers in our corner of the IT industry. Experienced speakers welcome anyone who wants to join them—they just caution you that it's hard, hard work.

OK, how about now? Still want to be a conference speaker? Start asking around. Your conference awaits!

Don Jones is a co-founder of Concentrated Technology (www.concentratedtech.com), helping to deliver IT knowledge in less time using innovative content techniques. He also serves as CTO and Series Editor for Realtime Publishers. Don is the author of more than 30 IT books, including Windows PowerShell: TFM; VBScript, WMI, and ADSI Unleashed; Managing Windows with VBScript and WMI; and many more. He is a multiple-year recipient of Microsoft's "Most Valuable Professional" (MVP) Award with a specialization in Windows PowerShell.

Greg Shields, MCSE: Security, CCEA, is an independent author, speaker, and consultant, based in Denver, Colorado. With more than 10 years of experience in information technology, Greg has developed extensive experience in systems administration, engineering, and architecture. Greg is a contributing editor for both Redmond magazine and MCPmag.com, authoring two regular columns along with numerous feature articles, webcasts, and white papers. He is also the resident editor for Realtime Publishers' Windows Server Community at www.realtime-windowsserver.com.

Practical PowerShell

Managing WMI Events with Windows PowerShell, Part 2

by Jeffery Hicks

You can download a zip file with all these scripts from http://www.realtime-windowsserver.com/code/v2n1_Practical_PowerShell.zip.

Last month, I demonstrated how to use PowerShell and Windows Management Instrumentation (WMI) to monitor events, such as the creation of a new file, change in a service, or the end of a process. These types of tasks can be handled by using a WMI notification query and the __InstanceCreationEvent, __InstanceModification, or __InstanceDeletion classes. This month, I'll show you another approach.

WMI has a few Win32 classes that you can use to monitor events related to Processes, Threads, and Modules. There is also a System trace class that includes all of these; although from an administrative scripting perspective, the Process trace classes are the one of most interest. There are WMI classes you can use to watch for when a process is started, Win32_ProcessStartTrace, or ended, Win32_ProcessStopTrace.

Like the Instance classes we looked at last time, you can't simply run a PowerShell command like this:

```
PS C:\> Get-WMIObject win32_processStartTrace -computername "XP02"
```

You won't get any errors, but you are also unlikely to see any results unless a process just happens to be starting at the same time. You need to keep your script "alive" so that it can receive event notifications. This is accomplished by using a ManagementEventWatcher object in much the same way as last month's scripts.

```
$namespace="\\$computername\root\cimv2"
 $query="Select * from Win32_ProcessStartTrace"
 $EventQuery = New-Object System.Management.WQLEventQuery $query
 $scope      = New-Object System.Management.ManagementScope $namespace
 $watcher    = New-Object System.Management.ManagementEventWatcher $scope,$EventQuery
 $options    = New-Object System.Management.EventWatcherOptions
 $options.TimeOut = [timespan]"0.0:0:1"
 $watcher.Options = $options
```

The query won't be executed until the Watcher object is started:

```
$watcher.Start()
```

To keep the script alive, I use a loop to wait for the next event to fire that matches the query:

```
#loop and wait for events to fire
 while ($true) {
    trap [System.Management.ManagementException] {continue}

 $evt=$watcher.WaitForNextEvent()
```

When an event fires, a Win32_ProcessStartTrace object will populate the $evt variable, which you can then write to the pipeline:

$evt | select @{name="Time";Expression={Convert-UTC $_.Time_Created}},ProcessName,ProcessID,ParentProcessID

The Win32_ProcessStartTrace object only has a few properties that I feel an administrator would be interested in, so I've piped the object to **Select-Object**. Here's a script you can use to watch for new process events on a remote computer.

```
#Get-NewProcessEvent.ps1
# Usage:
# Get-NewProcessEvent SERVER02

Param([string]$computername=$env:computername)


  Function Convert-UTC {

    Param([int64]$var=0)

    [datetime]$utc="1/1/1601"

    if ($var -eq 0) {
        write $utc
    } else {

        $i=$var/864000000000
        write ($utc.AddDays($i))
    }
}

 $ESCkey  = 27

 $namespace="\\$computername\root\cimv2"
 $query="Select * from Win32_ProcessStartTrace"
 $EventQuery = New-Object System.Management.WQLEventQuery $query
 $scope      = New-Object System.Management.ManagementScope $namespace
 $watcher    = New-Object System.Management.ManagementEventWatcher $scope,$EventQuery
 $options    = New-Object System.Management.EventWatcherOptions
 $options.TimeOut = [timespan]"0.0:0:1"
 $watcher.Options = $options
```

```
 cls

 #diplay a message about what the script is waiting for
 Write-Host "Waiting for events in response to: " $EventQuery.querystring "on
$computername. Press ESC to quit." -back cyan -fore black

 #start the management watcher
 $watcher.Start()

 #loop and wait for events to fire
 while ($true) {
    trap [System.Management.ManagementException] {continue}

 $evt=$watcher.WaitForNextEvent()

 #if an event fires....
 if ($evt) {
  #uncomment the next line if you want a beep every time a new process is started
  #write `a
  $evt | select @{name="Time";Expression={Convert-UTC $_.Time_Created}},ProcessName,Proce
ssID,ParentProcessID

  #clear the variable for the next event
  Clear-Variable evt
  }

 if ($host.ui.RawUi.KeyAvailable)
  {   $key = $host.ui.RawUI.ReadKey("NoEcho,IncludeKeyUp")
      if ($key.VirtualKeyCode -eq $ESCkey)
      {   $watcher.Stop()
          break
      }
  }

 }

#end of script
```

Like last month's script, it will keep looping until you press the Esc key, at which point, the Watcher object is stopped and the script terminated. As written, this script will display the time the process has started, the process name, its ID, and its parent ID. You could use these values to get more information by calling **Get-Process**. The timestamp is returned from a function that converts a UTC formatted string into a more meaningful value. But you'll notice that it is not your local time. I'll show you a solution for this a little bit later.

This script is intended more as a proof of concept, although it can be used as is. You can connect to a remote computer but the script doesn't support alternate credentials. The script also only monitors when new processes are created. But you might also want to watch when processes end. You could use the Win32_ProcessStopTrace or use the Win32_ProcessTrace class, which will cover process creation and termination.

The script, Get-ProcessEvent.ps1, which is available at http://www.realtime-windowsserver.com/code/v2n1_Practical_PowerShell.zip, is an extended version of my earlier script.

```
#GET-PROCESSEVENT.PS1

Param([string]$computername=$env:computername,
      [string]$filter="%",
      [int]$poll=10,
      [System.Management.Automation.PSCredential]$credential,
      [switch]$log,
      [switch]$beep
      )


Function Convert-UTC {
#returns a UTC date time
Param([int64]$var=0)

[datetime]$utc="1/1/1601"

if ($var -eq 0) {
    write $utc
 }
else {
    $i=$var/864000000000
    write ($utc.AddDays($i))
 }
}


 if ($log) {
   #if -log is specified a file will be created with a
   #name like C:\SERVER02_ProcessEvents_07192008193700.csv
   $d=Get-Date -f MMddyyyyHHmmss
   $logfile="C:\{0}_ProcessEvents_{1}.csv" -f $computername.ToUpper(),$d

   Set-Content $logfile "ComputerName,ProcessEvent,Time,ProcessName,ProcessID,ProcessPath
,ParentProcess" -encoding ASCII

   #define name for temporary csv file
   $tmpfile="$env:temp\~tmp$.csv"
 }

 #
```

```
get current time zone for the computer so that the UTC time can
 #be converted to local time

 if ($credential) {
  $tz=(Get-WmiObject win32_operatingsystem -computername $computername -credential
$credential).currentTimeZone
 }
 else
 {
  $tz=(Get-WmiObject win32_operatingsystem -computername $computername).currentTimeZone

 }

 $ESCkey  = 27

 $namespace="\\$computername\root\cimv2"
 $query="Select * from Win32_ProcessTrace Where ProcessName LIKE '$filter'"
 $EventQuery = New-Object System.Management.WQLEventQuery $query
 $scope      = New-Object System.Management.ManagementScope $namespace
```

```
 if ($Credential) {
  #use alternate credentials if passed
    $scope.options.Username = $credential.GetNetworkCredential().Username
    $scope.options.Password = $credential.GetNetworkCredential().Password
    $scope.options
  }

 $watcher    = New-Object System.Management.ManagementEventWatcher $scope,$EventQuery
 $options    = New-Object System.Management.EventWatcherOptions
 $options.TimeOut = [timespan]"0.0:0:1"
 $watcher.Options = $options

 cls

 #diplay a message about what the script is waiting for
 Write-Host "Waiting for events :" $EventQuery.querystring "on $computername. Press ESC
to quit." -back cyan -fore black

 #start the management watcher
 $watcher.Start()

 #loop and wait for events to fire
 while ($true) {

    trap [System.Management.ManagementException] {continue}

    $evt=$watcher.WaitForNextEvent()

     #if an event fires....
     if ($evt) {
        if ($beep) {write `a}

     #build a custom object showing what type of event, converted
     #date time values and other relevant information
     $data=$evt | select @{name="ComputerName";Expression={$computername}},`
     @{Name="ProcessEvent";Expression={
       Switch ($_.__Class) {
        "Win32_ProcessStartTrace" {"Start"}
        "Win32_ProcessStopTrace" {"Stop"}
        default {$_.__Class}
       }
       }},`
```

```
    @{name="Time";Expression={
      #convert time to UTC
      [datetime]$utcTime=Convert-UTC $_.Time_Created
      #adjust UTC to local time
      $utcTime.AddMinutes($tz)
      }},`
    ProcessName,ProcessID,`
    @{name="ProcessPath";Expression={
      $procPath=(Get-Process -Id $_.ProcessID).path
      if ($procPath) {
        $procPath
      }
      else {
       "N/A"
      }
      }},`
    @{name="ParentProcess";Expression={
    "{0} ({1})" -f $_.ParentProcessID,(Get-Process -id $_.ParentProcessID).processname}}

    #write process event data to pipeline
    $data

    # if -log was specified export data to the temp CSV file
    # then copy that content to the log file and remove the temp CSV file
    if ($log) {
      $data | Export-Csv -Path $tmpfile -encoding ASCII -NoTypeInformation
      Get-Content $tmpfile | select -Last 1 | Out-File $logfile -append -encoding ASCII
      Remove-Item $tmpfile
      }

    #Clear $EVT for next event
    Clear-Variable evt

    #sleep for the specified number of seconds
    Start-Sleep -Seconds $poll

} #end If ($evt)

#watch for ESC key
    if ($host.ui.RawUi.KeyAvailable)
```

```
      {   $key = $host.ui.RawUI.ReadKey("NoEcho,IncludeKeyUp")
          if ($key.VirtualKeyCode -eq $ESCkey)
          {   $watcher.Stop()

              break
          }
      }

} #end of while loop
```

This version of the script accepts parameters for a computername, a filter for a process name, a PSCredential for alternate credentials, a polling interval, plus logging and beep options. Here are some sample usages.

```
PS C:\> $cred=get-credential mycompany\admin
PS C:\> get-Process Event -computername "SRV01" -credential $cred
PS C:\> get-Process Event -computername "SRV01" -log
PS C:\> get-Process Event -computername "SRV01" -log -filter "notepad.exe"
PS C:\> get-Process Event -computername "SRV01" -log -filter "notepad.exe" -poll 30
```

If you specify the **–log** parameter, all output will be written to a file in CSV format. The script will create a filename based on the computername value and the date and time. For example, monitoring SERVER02 on December 19, 2008 at 7:37PM, might result in a filename like C:\SERVER02_ProcessEvents_1219200819370O.csv. Events will still be written to the screen in standard fashion. You can specify alternate credentials, but they must be in the form of a saved PSCredential object, as I previously showed.

If you specify **–beep**, your computer will beep once whenever a matching event fires. You might find this useful for long-running sessions where you want to minimize your PowerShell window but still be notified when an event occurs.

This script doesn't support the polling technique that I used with the Instance scripts from last month. But to even out performance, I've added a **Start-Sleep** command within the loop while waiting for the next event.

```
      Start-Sleep -Seconds $poll
```

The default is 10 seconds. So now what happens when a process related event fires?

First, if you specified **–beep**, your computer will.

```
      if ($evt) {
         if ($beep) {write `a}
```

The final parameter allows you to filter on a process name. By default, the WMI query will be:

```
Select * from Win32_ProcessTrace where ProcessName LIKE '%'
```

This is a wildcard query that will return objects where ProcessName has a value, which should be everything. However, you might only be interested in monitoring a specific process. You can use expressions like this.

```
PS C:\> get-Process Event -computername “SRV01” -log -filter “notepad.exe”
```

or

```
PS C:\> get-Process Event -computername “SRV01” -log -filter “%note%”
```

In other words, whatever value you specify as the filter parameter will be on the right side of the LIKE operator.

Instead of simply writing the event object, I’m going to create a custom object that is a little more meaningful. The custom object, $data, will have properties for the computername, and a property that indicates if the event was a process start or end.

```
    $data=$evt | select @{name=”ComputerName”;Expression={$computername}},`
  @{Name=”ProcessEvent”;Expression={
      Switch ($_.__Class) {
       “Win32_ProcessStartTrace” {“Start”}
       “Win32_ProcessStopTrace” {“Stop”}
       default {$_.__Class}
      }
      }},`
```

I use a Switch construct to compare the __Class property and set an appropriate value for the ProcessEvent property. Next, I define a Time property by taking the Time_Created value from the original object and reformatting it with the Convert-UTC function. Remember that this function returns a timestamp not necessarily in the local time. To adjust, I subtract the number of minutes pulled from the current time zone, which is retrieved earlier in the script.

```
if ($credential) {
  $tz=(Get-WmiObject win32_operatingsystem -computername $computername -credential
$credential).currentTimeZone
 }
 else
 {
  $tz=(Get-WmiObject win32_operatingsystem -computername $computername).currentTimeZone

 }
```

Here’s the code in the script that defines the Time property.

```
    @{name=”Time”;Expression={
      #convert time to UTC
      [datetime]$utcTime=Convert-UTC $_.Time_Created
      #adjust UTC to local time
      $utcTime.AddMinutes($tz)
      }},`
```

Of course, we need to know the Process Name and ID.

```
      ProcessName,ProcessID,`
```

Remember, the returned object is not an actual process object, so we don't have direct access to all of a process object's properties. But we can get them by calling **Get-Process** using the ProcessID property of the current object. I'll get the path property.

```
      @{name="ProcessPath";Expression={
        $procPath=(Get-Process -Id $_.ProcessID).path
        if ($procPath) {
          $procPath
        }
        else {
         "N/A"
        }
        }},`
```

For a process that has terminated, this property won't exist; instead, the custom object will simply display N/A.

The last property I want to display is the parent process. The Trace object will return the process ID of the parent process. Using **Get-Process** again, I can get the associated name.

```
      @{name="ParentProcess";Expression={
      "{0} ({1})" -f $_.ParentProcessID,(Get-Process -id $_.ParentProcessID).processname}}
```

I'm using the –f replacement operator to replace {0} with the parent's processed and {1} with the name of the parent process pulled from a **Get-Process** expression. The entire custom object is then written to the pipeline.

```
      #write process event data to pipeline
      $data
```

Before we wrap this up, let me quickly explain how I handled the logging feature. You could simply pipe the script to **Export-CSV**.

```
PS C:\> C:\scripts\Get-ProcessEvent.ps1 FILE02 | export-csv file02procs.csv
-noTypeInformation
```

Piping to **Export-CLIXML** or **Out-File** would also work. Although in these situations, you wouldn't see the output on the screen. So if specified, the –Log parameter instructs the script to pipe $data to **Export-CSV**. One limitation in the cmdlet is that there is no Append feature, so I create a temporary CSV file and write a header line.

```
   Set-Content $logfile "ComputerName,ProcessEvent,Time,ProcessName,ProcessID,ProcessPath
,ParentProcess" -encoding ASCII

   #define name for temporary csv file
   $tmpfile="$env:temp\~tmp$.csv"
```

Here's where the sleight of hand comes in. Using **Get-Content**, I read in the contents of the temp file and select the last line. This line is then piped to **Out-File**, which does have an append parameter. After that, I simply delete the temp file.

```
if ($log) {
  $data | Export-Csv -Path $tmpfile -encoding ASCII -NoTypeInformation
  Get-Content $tmpfile | select -Last 1 | Out-File $logfile -append -encoding ASCII
  Remove-Item $tmpfile
  }
```

In case you're wondering, I'm selecting the last line because I don't want to include the header.

As you've seen, it takes a little bit of work, but I hope these scripts give you a jump start on managing events with WMI and PowerShell. If you like these scripts, you'll love PowerShell v2.0 and its WMI event cmdlets. But that's a topic for a future column.

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

ActiveSync with SP1

by J. Peter Bruzzese

When a service pack comes out for a product like Exchange, we look for a few new features and wonder whether the development team listened to the needs of the administrators. The answer for Service Pack 1 (SP1) for Exchange 2007 is a resounding "Yes!" in terms of product enhancements. Keep in mind there was no reason to rewrite the entire structure; we just needed a few new things, such as a Public Folder management tool for the Exchange Management Console (which we got), a new high-availability feature (which we also received), and last but not least some additional ActiveSync policy settings (and we weren't let down is this regard by any means).

With RTM, we had 16 ActiveSync policy settings; with SP1, we have 27 settings under the Standard CAL and 43 settings for the Enterprise CAL. Not too shabby. Let's consider some of these settings.

ActiveSync Policy Settings with SP1

When you first look into the Organization Configuration work center under the Client Access node, you might expect to see nothing because the RTM had nothing in terms of a default policy in place. Not so with SP1. The default policy can help you get started, or you can create your own policies from the Actions pane.

After creating the policy, you can enter the properties, and you will be presented with five tabs: General, Password, Sync Settings, Device, and Advanced. The General tab is similar in SP1 to its predecessor with the one odd option Allow non-provisionable devices. I say odd because that isn't really a word according to my spell check. If you select this option, though, it will allow older devices to connect to EAS even if all the settings will not apply. Great for legacy

devices; not so great for enforcement of modern policy settings. So, you might want to turn off this setting if you have confidence that your users have mobile devices that will work under the policy settings you choose.

> Note: As for mobile devices having the ultimate in policy application ability, you should be looking to provide users with Mobile 6.1. Otherwise, the majority of the new policy settings will not apply.

The Password tab has a few welcome improvements in SP1. In addition to settings such as password length, password expiration, and so forth, we now have a Minimum number of complex characters setting, which is any character that is not a letter. We also have a Require encryption on the storage card setting, which again is not supported on all devices, so make sure your device supports your policy requirements.

Sync Settings offers options that relate to past calendar and email items that you want to have in sync with the device. You can limit the download size of messages, determine whether you want synchronization to occur while roaming, allow/disallow HTML formatted email, and/or Allow attachments to be downloaded to the device (which is not a feature that is new to SP1; it was in the RTM on the General tab and has just received a new location in the dialog box).

As for the Device and Advanced tabs, these are special not only in that they are completely new to SP1 but also in that they are meaningless to a mobile device that doesn't have the latest flavor (Mobile 6.1) and doesn't have premium features enabled (which is only for Enterprise Client Access License holders).

That being said, Figure 1 highlights the following features that you can enable or disable via the Device tab:

- Allow removable storage
- Allow camera
- Allow Wi-Fi
- Allow infrared
- Allow Internet sharing from the device
- Allow remote desktop from the device
- Allow synchronization from a desktop
- Allow Bluetooth (with options to Disable, Handsfree Only, and Allow)

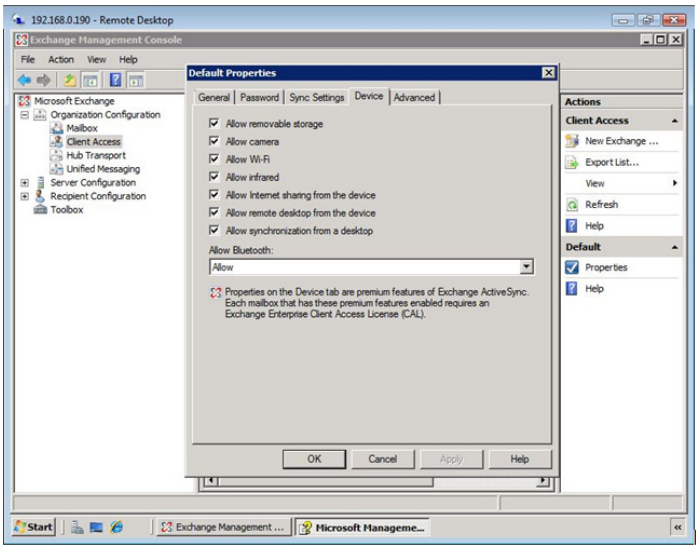

Figure 1: The Device tab of the default ActiveSync policy with SP1.

The Advanced tab (see Figure 2) continues the enable/disable structure with options such as Allow browser, Allow consumer mail, Allow unsigned applications, Allow unsigned installation packages, and Allowed Applications and Blocked Applications.

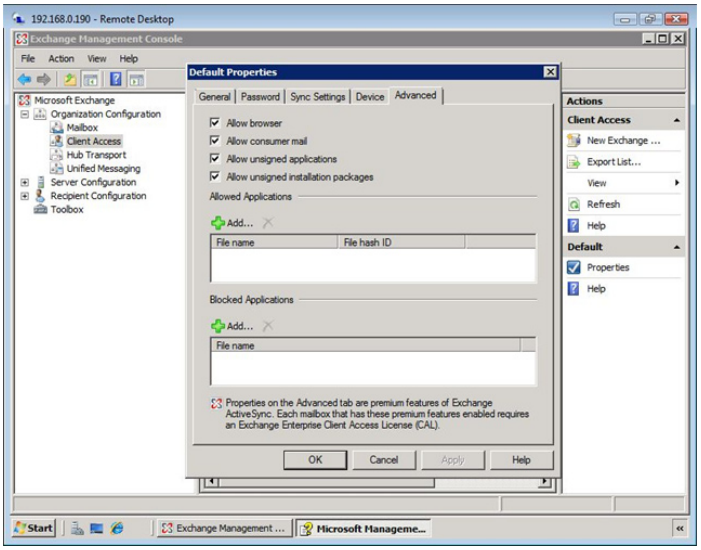

Figure 2: The Devices tab of the default ActiveSync policy with SP1.

Beyond ActiveSync

The Microsoft Exchange Team is doing a fine job of incrementally enhancing the Exchange feature set and we appreciate it. That doesn't mean, however, that we have to push BlackBerry out of the picture. A BlackBerry Enterprise Server can work in harmony with your Exchange server if you feel you want to work with another type of mobile

device. Then again, if you decide you want to keep an 'all-Microsoft-shop,' that is ok too, but you might need more than what ActiveSync can offer. In that case, you should look into the System Center Mobile Device Manager (MDM), which takes the concept of mobile device policies to an entirely new level.

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: ActiveSync Properties (Running Time: 2 minutes, 24 seconds), ActiveSync Policies from A to Z (Running Time: 5 minutes, 20 seconds), and Manage Your Mobile Device From Outlook Web Access (Running Time: 2 minutes, 5 seconds).

Copyright Statement