Realtime publishers

August 2009
Volume 2, Number 8

Windows Administration
in Realtime

Letter from the Editor

Wither Telecommuting?

by Greg Shields

I had a thought while walking through Microsoft's Tech Ed expo floor this spring. I'm walking around the massive show floor, watching software company after software company huck their wares and tell me why their product is The Next Big Thing. Many of those groups dealt with solving the problems of remote access:

> "Use our networking device to speed your remote access!"
>
> "Our software package makes remote access just like being in the office!"
>
> "Consider our automation tools for making the administration of remote access easy!"

In walking past all these companies, the thought struck me: We in IT spend an inordinate amount of time building and managing systems that enable our workforce to work from home or on the road. But rare is the IT professional who themselves is a telecommuter. We've got remote access solutions. We've got Terminal Services. We've got automation studios that enable massively-widespread administration of desktops, laptops, and servers. We've even got VoIP phone solutions that could pull our desk phone to our cell or home phone.

But why aren't we telecommuting?

Is it because we've historically been the people who need to be on-site? Do we need to be around just in case the servers fail? Even with our remote control capabilities, must we actually be a belly button for someone to physically ask questions?

To me, the IT career path seems like one that's ripe for gas-cost avoidance. Telecommuting at first blush appears to fit perfectly into the mindset, technology, and goals of IT. But most of the people I talk with still aren't doing it, or aren't allowed to do it.

What about you? Does your IT job involve some measure of telecommuting? Was it hard to get approved or is "keeping people off the roads" a priority at your company? I'd love to hear about it. Send your thoughts to gshields@realtimepublishers.net.

Break the Habit

Role Loading

by Don Jones

I used to work on F-14s for the Navy, and every Navy carrier had a jet they called the "Hangar Queen." This was the jet nobody flew: It just ruled over the hangar deck, contributing spare parts to jets that needed them. When I moved into IT, I started seeing servers that I nicknamed "Rack Queens." Not because they were used for spare parts, but rather because they commanded the most attention, maintenance, and worry from administrators. These servers were being used for a poor practice I now like to call *role loading*.

Here's an example: I had an NT 4 (I'm showing my age, I guess) backup domain controller that also ran the RightFax fax server software, was connected to a couple of external CD towers (thus making it a file server), served up printer queues, and oh—by the way—was the primary Exchange Server 5.5 box in he environment. Yikes!

Role loading is when a single machine starts to take on multiple unrelated roles. It creates major headaches: for example, patching the various software applications, as you're worried about securing them all and you know that a single bug in a single app can make the entire server—and all the services is offers—unstable. I don't think any admin *likes* role loading or sets out to do so, but sometimes we get stuck between a rock and a hard place. You *must* have a particular service added to the network, and you *can't* buy new hardware; what, besides role loading, are you supposed to do? Today, there are two approaches for avoiding this problem. You can actually use these together as complementary ways to help protect your infrastructure.

The first is to start using Windows Server 2008's Server Core installation option. This feature lets you build dedicated infrastructure servers for Active Directory (AD), DNS, DHCP,

and so forth. By using Server Core, you place a hard limit on the roles the server can perform. Although it is still possible to load a server with too many roles, it becomes a lot harder to add major external roles such as Exchange Server, SQL Server, fax servers, and so forth because those products *won't run on Server Core*. Server Core thus helps establish a maximum role workload for each server, letting you protect your critical infrastructure servers from the rock-and-a-hard-place decisions.

The second is to utilize virtualization—VMware, Hyper-V, Xen, or whatever you like. Virtualization lets you run many roles on a single piece of *hardware* but helps to avoid the problems normally associated with role loading by giving each major role a separate operating system (OS) and virtual hardware environment. This is a pretty obvious solution for most of us these days, but there's a subtle point to be made: Companies are still concerned about *virtual sprawl*. Virtual machines are not maintenance-free; they still require OS patches and other maintenance, so just spinning up a new virtual machine for every new role isn't practical. What you can do, however, is size the virtual machines to limit them to a designated set of roles. For example, I love to virtualize my Server Core infrastructure servers, and I allocate them drive space, memory, and other resources for that infrastructure role. Adding more roles in the future isn't impossible—most virtualization products allow you to reallocate resources fairly easily—but it does create extra steps and helps push me toward the "make a new virtual machine" decision, especially for major roles like messaging, databases, collaboration, and so forth.

The moral: Avoid role loading. You can make your servers perform better, keep them easier to manage, and help keep yourself more efficient by separating roles. With today's technologies, you can do so with a lot less hardware sprawl than in the past, and options such as Server Core help enforce and protect your decisions, especially for the most critical network roles.

Share your own "worst practices" with Don by asking a question at his Web site, www.ConcentratedTech.com.

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.

Product Review

PortableApps Suite

by Eric Schmidt

Maintaining privacy in the Internet age has become increasingly difficult, while at the same time more aspects of daily life have become dependent on it. Often there are times where one may need to check email, bank accounts, or print a boarding pass from a public computer in a hotel or Internet café. Individuals that work in IT also rely heavily on both the Internet and intranet to perform their daily tasks and support their customers.

The PortableApps suite from PortableApps.com can address both concerns by providing a host of applications that can be customized and run on any Windows computer. From a privacy perspective, the applications run from a portable device and as such everything that is done is only stored on that device and not on the computer where it's running. For IT support staff, systems administrators, and consultants that support users and their computers, the portable applications provide access to a toolkit of applications, utilities, passwords, and documentation no matter what system they're using.

The standard edition of the suite includes portable versions of Firefox (browser), Thunderbird (email), Sunbird (calendar), ClamWin (anti-virus), Pidgen (instant messaging), Sumatra (PDF reader), KeePass (password safe), and OpenOffice. It also includes two games (Sudoku, Mines-Perfect) as well as the CoolPlayer+ audio player. All the applications are integrated into an easy-to-use interface that looks similar to a Windows menu. The standard suite requires 355MB of free space, so with the relative low cost of 2GB and 4GB flash drives, the suite will leave plenty of room for other files. Best of all, the entire suite of applications consists of free, open source software.

Installation of the suite is very straight forward. After downloading the 127MB file, the installer can be pointed directly to a portable device. Once installed, the suite is opened by running the base executable or, if autorun is enabled, it can be launched as soon as the drive is connected. From there, all the applications are available and ready to be used. The application list is not limited to what's provided; the interface has the ability to add links. With this functionality, one can build a customized toolkit that includes other utilities and applications that may be needed for troubleshooting and support. In fact, the PortableApps Web site has many additional open source applications and utilities available.

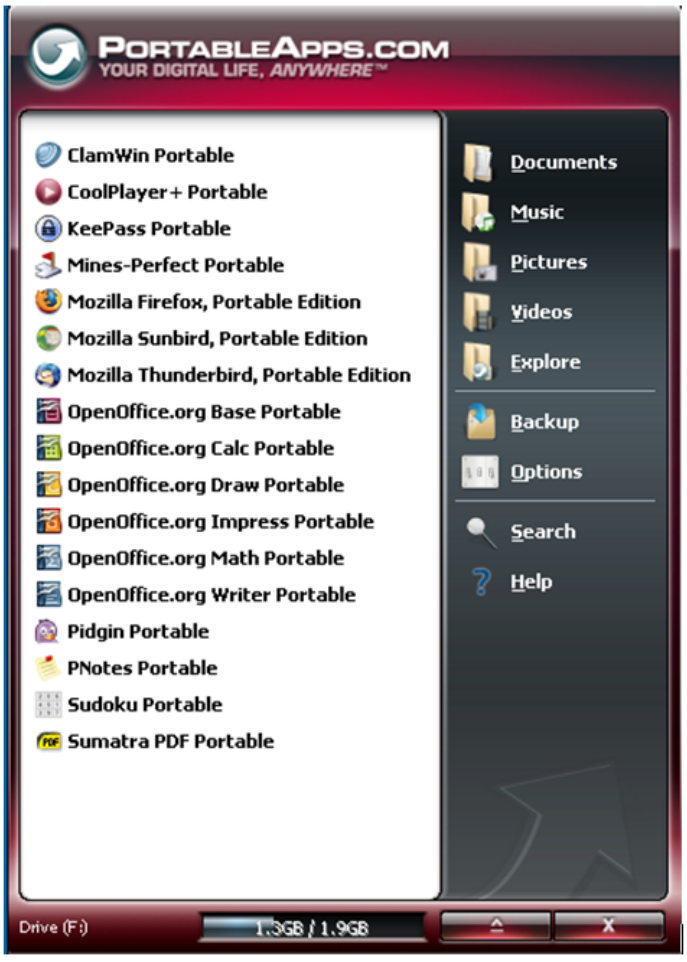


Figure 1: The application interface.

In large enterprises, the suite can be leveraged to store links to support documentation, which is then available with the portable browser and PDF reader. This feature can decrease the time spent searching for resources while supporting systems in the field. When working on a customer's computer, the portable suite can be used to access resources such as passwords or Web sites that the user may not need to have available to them after the support work has been completed. Performing these tasks using the portable apps instead of the installed browser keeps all the activity out of their browser history.

Overall, the PortableApps suite offers a wide array of applications that establishes a solid foundation for a portable IT toolkit. This kit can be used effectively by consultants and IT personnel in both large and small environments.

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.

Compliance Is Not Intimidating!

by Mitchell Thompson

Think positively. Regulatory compliance should be viewed as getting credit for all the work that most Windows administrators already perform. Specific frameworks may emphasize certain areas more than others, but as standards are becoming more established, tools for compliance are often included within operating systems (OSs) or resource kits. Thankfully, published standards are known around the world and can be used as lenses to view technology management based on particular risks.

This article describes how the Health Insurance Portability and Accountability Act (HIPAA), Payment Card Industry Data Security Standard (PCI-DSS), Sarbanes-Oxley (SOX) Act, and Gramm-Leach-Bliley Act (GLBA) rules emphasize controls surrounding different areas of technology management. There are universal responsibilities across each approach, such as limiting user account access, protecting server hardware, backing up critical data, and monitoring event logs. Each standard has a specific focus (for example, credit card data, financial processing, customer information) and requires organizations to perform a risk assessment in order to identify which servers host sensitive data. Of course, access to read and write to protected data should be restricted only to those with a defined need to do so.

To prove such is the case, IT auditors want to know that several layers of controls are in place to ensure the effectiveness of protection. To demonstrate to an auditor that information is appropriately restricted, Windows administrators need to show a combination of preventative, detective, and corrective controls. Examples representing each type of controls include:

- Preventative—Security groups limited to authorized user accounts
- Detective—Logging of events and alerting of suspicious activity
- Corrective—Quarterly review of security group membership

Comparing different standards outside of simple truisms often leads to inconsistencies in information. This reality makes sense when you consider the original intent of each framework, but it can be frustrating to systems administrators. The following sections offer highlights from each standard to show how approaches can vary and what tools are commonly used as part of compliance.

HIPAA

Companies that process healthcare and patient information must protect their data in accordance with the HIPAA Security Rule. What a Windows administrator needs to know about HIPAA is that its requirements are categorized as either required or "addressable," which means that organizations can assess whether such a safeguard provides additional value before implementing the safeguard. One example of a required safeguard is "Device and Media Controls," which mandates a formal process around the decommissioning of hard drives containing personally identifiable healthcare information. Cipher.exe has been shipped with Microsoft server OSs since Windows 2000 Server and can be used to sanitize drives before they are disposed of. Examples of "addressable" controls include password management and automated account logoff, which understandably can vary from organization to organization and are configurable entirely through Group Policy.

PCI-DSS

Preventing unauthorized exposure of cardholder data is the focus of PCI-DSS. Windows administrators with environments that process limited credit card transactions can perform a self-assessment of the 12 requirements for PCI-DSS compliance. There are four merchant levels that companies are categorized as under PCI-DSS; these levels are based upon the volume of transactions processed. For companies with more than 20,000 e-commerce transactions a year, a third-party may be required to conduct vulnerability and penetration testing from the Internet every quarter. Disabling unnecessary services using Internet Information Server lockdown tools and ensuring patch levels are current with the Microsoft Baseline Security Analyzer can minimize the number of issues identified by these scans. PCI-DSS explicitly requires protecting data across public networks by encrypting the transmission of cardholder data using modern encryption methods. PCI-DSS also requires that default user names and passwords not be used. This requirement can be addressed for Windows servers using the Group Policy "rename guest account" and "rename administrator account" settings or by disabling the accounts entirely.

SOX

SOX applies to publicly-traded companies and ensures financial transactions are both authorized and properly processed. The goal for Windows administrators is to show reduced risks of automated data processing errors to auditors by following well-designed formal processes that can be tested. For example, to help ensure controlled access, Human Resources may provide a list of employee changes over a period of time; for each employee that is hired, transitions to a different position, or is terminated, evidence must be provided that the employee's account permissions were edited appropriately. Many Windows administrators log details for each change received with Help desk requests and move disabled accounts to an Organizational Unit (OU) in Active Directory (AD) so that auditors can test account attributes long after access has been revoked. Windows administrators with a formal process of software change controls can show reduced risk of data processing errors by keeping evidence of software testing and approval prior to implementation of changes to production. Virtualization has made this process remarkably easier by allowing administrators to copy production systems into an isolated lab environment where testing can be conducted.

GLBA

The GLBA Security Guidelines and Privacy Rules state how financial institutions must protect their customer data. GLBA rules outline many internal controls but also specifically state that external service provider access must be actively monitored. Connectivity with third parties complicates account administration because administrators are not always informed of changes in staffing by partner organizations. Whenever feasible, shared user accounts should not be assigned to outside parties and additional precautions such as restricted logon times, locked down terminal server environments, or simply disabling the account when it is not in use should be considered. Hundreds of similar safeguards are configurable through AD and Group Policy.

Compliance Tools Abound

Ultimately, compliance with regulatory frameworks becomes another tool available to Windows administrators. Keeping evidence and making time on a regular basis to revisit controls and double-check whether assumed protections are in place may identify areas that have not received the attention they deserve. At the same time, reporting of compliance reminds managers of all the background tasks required to properly run a modern environment.

There are certainly penalties from regulators and risks to sensitive information associated with non-compliance to consider, but scrambling to buy compliance tools may not be necessary. For example, many companies maintain sophisticated log monitoring applications when integrated technologies such as scheduled WMI scripts or Microsoft's EventCombMB tool may address the need. As compliance needs of Windows administrators become more established, more built-in tools will become available to benefit the technology community, no matter what standard is applied. ◆

Mitch Thompson has worked in highly regulated industries such as the armed services and public accounting for 10 years. He currently works as a business consultant specializing in mid-sized public and private companies. His top three favorite subjects to write about are sensible auditing, simplifying IT processes and quantifying IT issues. Mitch lives in Seattle, WA with his wife Rebekah.

The Deep Dive

The "Officially Correct" Process to Use ESEUTIL & ISINTEG Against a Busted Exchange Database

by Greg Shields

For this month's Deep Dive, I present you with the #1 most popular page found in Realtime Windows Server Community. This post has for more than 2 years scored the highest number of page hits month-over-month.

Recognizing its value, I reprint it below in its entirety in case you may have missed it. If you want more great tips and tricks just like this, check out the Realtime Windows Server Community at www.realtime-windowsserver.com. There, you'll get a new post every day serving up all the great Windows Server topics, trends, and technology you're asking for.

I was called to a client company last week to help them with an Exchange 2003 database that had dumped its brains and would no longer mount either private or public store. Because of the way in which the server had lost its brains, it was not possible to complete an ESEUTIL /R to recover the database. A much-more scary ESEUTIL /P was needed to do a repair.

In researching the exact procedure necessary to fully complete the repair, I found quite a bit of differing information about which of the steps were actually necessary for the repair. Some newsgroups said the ESEUTIL /P was

the only step necessary. For others, a multi-step process was necessary. In an attempt to be thorough (and having an available no-cost PSS case available), I contacted Microsoft to get the correct fix. This is their official response.

Step 1: Repair the Database

Run *Eseutil /p {Path to EDB file}*

This process takes about an hour per 8GB of database size. If you haven't partitioned your Exchange stores into multiple, smaller stores, this process will have you saying, "I should have done that" before it's fully done.

Note that although some very nice progress graphs will appear to show you how far it has left to go, I have found that it hangs by far the longest at the step of "deleting unicode fixup table" which has no progress graph. DO NOT stop the process if it hangs here. Let it complete.

Step 2: Defrag the Database & Rebuild Indices

Run *Eseutil /d {Path to EDB file} /t {Path to Temp EDB file}*

You must have an additional 110% free space on the actual drive where this will occur or the process will not complete properly, leaving you to re-try and wait longer. This 110% includes the STM file size as well! This process can also take up to an hour per 8GB of database size.

If you do not have enough space to complete this task on the Exchange server, you can move the database files to an alternate location and run the defragmentation there. To do this, copy the files Eseutil.exe, Ese.dll, Jcb.dll, Exosal.dll, and Exchmem.dll files from the Exchange Server 2003 to the remote location along with the Exchange Database (edb & stm). Then, run ESEUTIL /D from the remote location. When complete, move the completed files back. Note that /t is the path to the Temp Drive with the 110% available free space.

Step 3: Fix Removed Linkages

Run *isinteg -s servername -fix -test alltests*

You will need to run this third command may times until the result is "0 Error and 0 Fixes". If you do not get this result after the first passthrough, continue running it until the result is correct. The ISINTEG command will only be able to fix a certain number of problems per run-through. So, many times may be necessary, depending on the level of corruption. If possible, also try to fix any warnings as well, depending on the time you have available. This command will take about 20 minutes to complete, depending on database size, per run-through.

Have thoughts on this tip? Comment on it at http://www.realtime-windowsserver.com/tips_tricks/2007/03/the_officially_correct_process.htm.

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

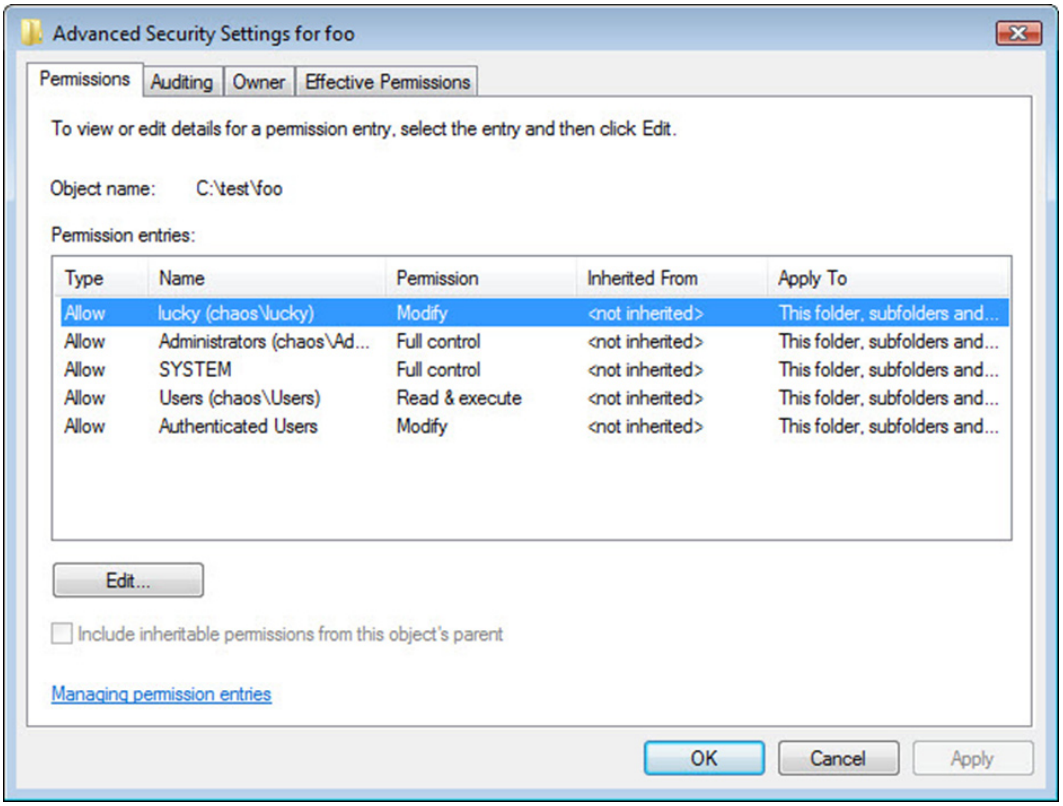
Protected Inheritance

by Jeffery Hicks

You can download a zip file with all these scripts from http://www.realtime-windowsserver.com/code/v2n8_Practical_PowerShell.zip.

I love spending time in scripting forums; you'll often find me as SAPIENScripter. I love it not only because I enjoy helping and educating but also because I get ideas for many of my columns and scripts, like the one I have for you this month. Because I no longer work in a large enterprise, there are some tasks I simply don't think about, such as managing file and folder permissions.

The challenge I assisted with was to search a directory structure and identify folders where security inheritance had been disabled.



This setup has the effect of protecting the access control list (ACL) for a specific folder. The task was to develop a PowerShell solution to identify these folders so that someone could evaluate them and determine which ones needed to be modified. Here's my PowerShell script solution.

```
Function Get-ProtectedACL {
    Param([string]$path=$(Read-Host “Enter a starting path”),
          [switch]$recurse,
          [switch]$unprotected,
          [switch]$verbose
          )

#capture the date time which will be used at the end
#to calculate how long the function took to execute.
$start=Get-Date

#turn on Verbose pipeline if -verbose
if ($verbose)
{
    $VerbosePreference=”Continue”
    Write-Verbose “$(get-date -format ‘MM/dd/yyyy hh:mm:ss:ffff’) Starting Function”
}

Write-Verbose “$(get-date -format ‘MM/dd/yyyy hh:mm:ss:ffff’) Verifying $path”

#Verify path exists
if (Test-Path $path)
{
    Write-Verbose “$(get-date -format ‘MM/dd/yyyy hh:mm:ss:ffff’) $path verified”

    if ($recurse)
    {
        Write-Verbose “$(get-date -format ‘MM/dd/yyyy hh:mm:ss:ffff’) -recurse was
specified”
        $cmd={Get-ChildItem $path -recurse}
    }
    else
    {
        $cmd={Get-ChildItem $path}
    }

    Write-Verbose “$(get-date -format ‘MM/dd/yyyy hh:mm:ss:ffff’) `$cmd is $cmd”
} #end if item exists
else
{
    Write-Verbose “$(get-date -format ‘MM/dd/yyyy hh:mm:ss:ffff’) Displaying error
message”
```

```
    Write-Warning "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff') Failed to find $path"
    return $null
}

Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff') Executing main command"
 &$cmd | where {$_.PSIscontainer} | foreach {
      Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff') Calculating Protected
property on $($_.fullname)"
      $protected=(($_ | get-acl).AreAccessRulesProtected)
      Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff') Adding Protected
property to $($_.fullname)"
      $_ | Add-Member -MemberType Noteproperty -Name "Protected" -value $protected

      #send appropriate objects down the pipeline
      if ($unprotected -and -not $_.protected)
      {
        Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff') -unprotected passed
and folder $($_.fullname) is not protected"
        write $_
      }
      elseif ($unprotected -and $_.Protected)
      {
        Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff') Skipping $($_.
fullname) because it is protected"
      }
      elseif ($_.protected)
      {
        Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff')$($_.fullname) is
protected"
        write $_
      }

      #clear $protected in case we hit a folder we can't read
      Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff') Clearing Protected
variable"
      Clear-Variable protected

    } #end foreach

#capture date and time
$end=Get-Date
```

```
#calculate total processing time
$processingTime=$end-$start

Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff') Ending Function"
Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff') Total time
$($processingTime)"

} #end function
```

I developed the function so that you could use it like a cmdlet. Click here to download the script file and dot source the function into your shell or script:

```
PS C:\Scripts\> . .\get-protectedacl.ps1
```

The function requires a path to analyze. If you don't include this parameter, you will be prompted:

```
    Param([string]$path=$(Read-Host "Enter a starting path"),
```

The function also takes parameters that specify whether to search recursively from the designated folder, whether to find unprotected folders (that is, those with the check box selected), and whether to enable verbose output. This last parameter, -verbose, will turn on the Verbose pipeline:

```
if ($verbose)
{
    $VerbosePreference="Continue"
    Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff') Starting Function"
}
```

Throughout the script, I have **Write-Verbose** expressions to provide informational messages regarding the function's progress.

Assuming the folder exists, which I verify using **Test-Path**, the function defines a script block to execute **Get-ChildItem**:

```
    if ($recurse)
    {
        Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff') -recurse was
specified"
        $cmd={Get-ChildItem $path -recurse}
    }
    else
    {
        $cmd={Get-ChildItem $path}
    }
```

The scriptblock is executed, filtering out anything that isn't a folder. Each folder object is piped to **ForEach-Object**:

```
&$cmd | where {$_.PSIscontainer} | foreach {
```

My plan is to add a new property to the folder object that will show whether the folder has a protected ACL. I can use **Get-ACL** and check the AreAccessRulesProtected property. So as part of the ForEach loop, I define a variable that will capture this property:

```
      $protected=(($_ | get-acl).AreAccessRulesProtected)
```

I'll add this property to the current folder object by piping it to **Add-Member**:

```
      $_ | Add-Member -MemberType Noteproperty -Name "Protected" -value $protected
```

All that remains is to determine whether the object should be written to the pipeline. Remember, the primary usage will be to only display folder objects where $protected will have a value of $TRUE. However, there may be situations where I want to find the opposite. That's why I added the –Unprotected parameter. I use an If..ElseIf construct to determine when to write $obj to the pipeline. The Write-Verbose commands will help you understand when an object is written to the pipeline and why:

```
      #send appropriate objects down the pipeline
      if ($unprotected -and -not $_.protected)
      {
        Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff') -unprotected passed
and folder $($_.fullname) is not protected"
        write $_
      }
      elseif ($unprotected -and $_.Protected)
      {
        Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff') Skipping $($_.
fullname) because it is protected"
      }
      elseif ($_.protected)
      {
        Write-Verbose "$(get-date -format 'MM/dd/yyyy hh:mm:ss:ffff')$($_.fullname) is
protected"
        write $_
      }
```

That's basically it. The function will display how long it took to run if you use –verbose.

What happens when you try to run the function?

```
PS C:\scripts\posh> get-protectedacl c:\test


    Directory: C:\test


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----          4/16/2009   9:09 AM            foo
d----          4/24/2009  11:20 AM            foo2
```

These two subfolders have the inheritance checkbox cleared, so they are the only ones written to the pipeline. You won't see the Protected property because the default view doesn't know what to do with it. But you can see it if you pipe the above expression to **Get-Member**. Or simply specify the property name:

```
PS C:\> Get-ProtectedACL c:\test -verbose -recurse | format-table fullname,protected
-autosize
```

Perhaps you'd like to create a text file with full path names for any folder with a protected ACL that you could use later:

```
PS C:\> get-protectedACL c:\files -r |  foreach {
  $_.fullname | Add-content c:\protected.txt }
```

The script should work with any NTFS-based file system drive, even a mapped network drive. As long as you can use Get-ACL, this function should work. But what about modifying these folders to unprotect them? We'll save that for another day.

If you run into problems with this script or have suggestions for improvements, please let me know in the PowerShell forum at ScriptingAnswers.com. ◆

Jeffery Hicks (MCSE, MCSA, MCT) is a Microsoft PowerShell MVP and Scripting Guru for SAPIEN Technologies. Jeff is a 17 year IT veteran specializing in administrative scripting and automation. Jeff is an active blogger, author, trainer and conference presenter. His latest book is Managing Active Directory with Windows PowerShell: TFM (SAPIEN Press). Follow Jeff at Twitter.com/JeffHicks and blog.sapien.com. You can contact Jeff at jhicks@sapien.com.

Tips and Tricks of the Exchange Management Shell

by J. Peter Bruzzese

I can remember the days of the Commodore 64. Sitting for hours, DAYS even, to type in a game from a magazine using Basic with strings and so forth only to see a ball bounce around the screen and change colors. My largest programming success came when I was able to convert a Blackjack game that didn't allow betting into a casino royale with a $5000 initial starter and the ability to double down.

Time moves on and the GUI eventually replaced the command line in the Windows world and I never looked back. That is, until I started playing around with PowerShell. PowerShell is more than functional and beyond powerful—it is actually enjoyable. Pipelining cmdlets together like an assembly line that takes parts and eventually produces a ca is exciting if you know how to do it. In this month's column, I'll share a few interesting tips and tricks about the Exchange Management Shell (EMS) that form a potpourri, an EMS cornucopia if you will, of interesting facets.

Get and Set Pipelines

Typically, you want to make changes fast through the EMS shell and those changes will (also typically) relate to mailboxes. So to start with, you need to Get what you need. Type

```
Get-Mailbox
```

and you get all mailboxes, but type

```
Get-Mailbox -Server <ServerName>
```

and you only get those mailboxes on that particular server. Narrowing your scope, or *filtering*, is important before you pipeline off to the Set cmdlets. Once you have the filtered scope, you can then Set-Mailbox, as the following example shows:

```
Get-Mailbox | Set-Mailbox -prohibitsendquota 20MB
```

The end result is an organization-wide change that alters the prohibitsendquota to 20MB for all mailboxes. All in one line. That is the PowerShell motto: Changing the world, one-liner at a time.

You can also try other types of Get options such as Get-User rather than Get-Mailbox. And you can use the –filter parameter in a manner such as this:

```
get-user -filter "Department -like '*Shipping*'"
```

which will search for users that are in the Shipping department only. At that point, you can pipeline it over to a Set-Mailbox cmdlet to follow through.

My recommendations is that you do your best to work through as many different possible GET/SET scenarios as you can because you are most certainly going to be needing them in the future. As a little tip, you should also search for Get-MailboxStatistics, which is yet another great cmdlet.

Standard or Enterprise Edition

I honestly cannot tell you how many times I went to perform a task on a server and didn't have the correct edition installed. How is one to know ahead of time the evaluation they have installed on their servers? You might try the

```
Get-ExchangeServer <ServerName> | Format-Table Name, Edition
```

to see the edition on a particular server. Actually, if you wanted to see all servers in your organization, you could simply leave out the <ServerName> portion.

However, an even better solution is a free script you can find at Expta.com (http://www.expta.com/labels/PowerShell.html) where the output will display the server name, Exchange roles installed, version (Standard or Enterprise), version number, and the Update Rollups installed and their installation dates. Incredible stuff.

ForEach for Redundancy

I saw this option in one of the EMS tips that pop up from time to time. It catches your attention because it isn't a cmdlet but it uses cmdlets within the line to create multiple items. For example, note the two lines of code below:

```
foreach ($i in (3,4)) {new-storagegroup -name SG$i -server "Server1" -logfolderpath "C:\
mounts\SG$i\Log" -SystemfolderPath "C:\mounts\SG$i\log"}

foreach ($i in (3,4)) {new-mailboxdatabase -storageGroup Server1\SG$i -Name MDB
-edbFilePath "C:\mounts\SG$i\db\MDB.edb"}
```

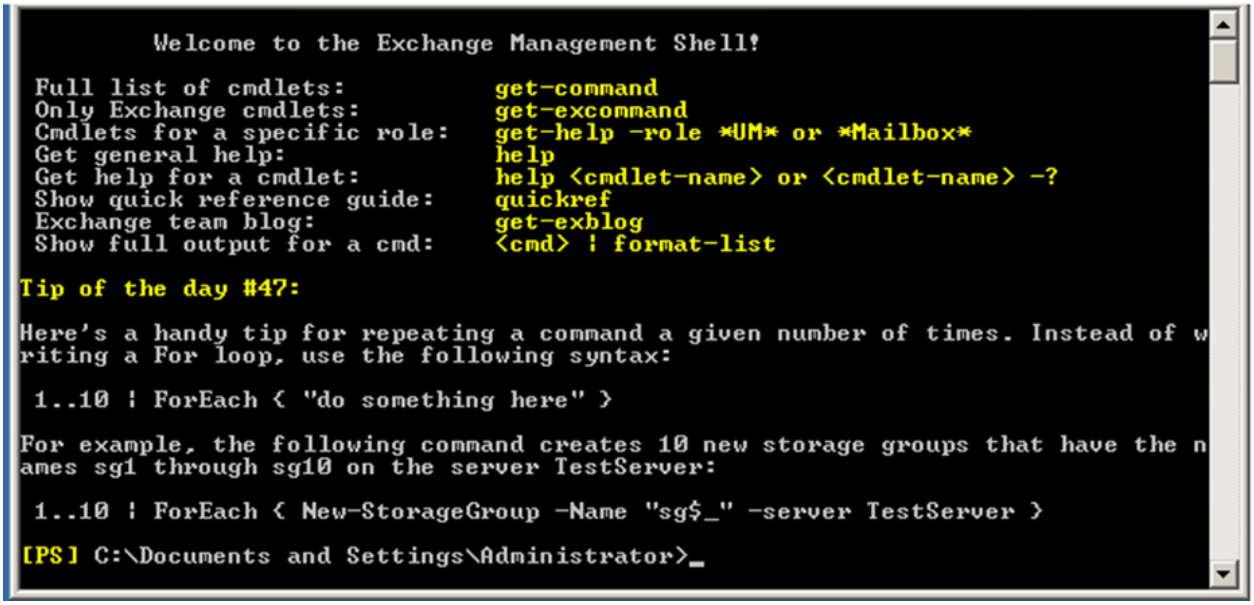


Figure 1: The EMS with Tip #47 ForEach.

I do a lot of testing with Exchange and need to build and rebuild Exchange Servers a lot (too much!). Notice the first line of code. It will create two storage groups named SG3 and SG4. You can use this trick for as many as 50 storage groups, of course, but it was easier to show just two. The second one takes those storage groups and creates databases inside of them. Note that the databases use the same name of MDB.edb, but you can change that as well.

How to Create a PowerShell Script

Scripts have a PS1 extension, and you will find a bevy of freely available ones in your Exchange bin folder on a server on which you have installed Exchange. Seriously, if you haven't seen these yet, you need to open the folder because you will locate some important ones:

- Install-AntiSpamAgents.ps1
- ResetSearchIndex.ps1
- Move-TransportDatabase.ps1

However, if you wanted to create your own scripts, perform the following steps:

1. Create a script by entering Exchange Management Shell commands in a text editor.
2. Save the text file for the script with the .ps1 file extension.
3. At the command prompt of the EMS, enter the path and file name of the script.

It really is as simple as that. But you don't have to be the creator of scripts, you can locate them online for free and use them to do some fantastic things with PowerShell. Oh, sure, it's hard to imagine writing a script that will play Blackjack and allow you to double down… but you never know. I'd certainly be impressed by anyone who could get that to work. Until then, we can simply enjoy managing our Exchange environment with one of the most flexible and powerful utilities we have been given to date to accomplish the task.

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. His most recent book "Exchange 2007 How-To" was published by Sams in January 2009. 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.

Copyright Statement