The fine art of Yak Shaving

I'm often asked just what it means to be "yak shaving". I can't tell you what the true origin of the phrase is, but believe me, I can tell you that I have shaved plenty of yaks in my career.

Hold still there little buddy, this may take a while

Let's take the example of a process that I'm working on which is to take a CSV file of data which is needed to create and update Microsoft Exchange 2010 contact records. It doesn't seem like a big deal, right?

With the Exchange Management Shell it would appear that the world is my oyster. Unfortunately, I find very quickly that I have some hurdles to get over.

Enter the yak.

Problem #1 – The data that I receive is in a format that isn't compatible with the raw data formatting for the native Exchange contact fields. When I import the fields, I have to concatenate a number of fields such as address (in 3 parts, with 1, 2 or 3 fields populated) and phone number (some numbers are international which creates different formatting) and to top it off the names include a "formal" and "informal" name, where we need the informal name if it is present, but otherwise use the formal name. This is more of a logic issue than a true yak though.

Problem #2 – You can't create a Contact record with all fields, but you can't modify the additional fields unless the initial contact exists. I now have to do a check for the contact to see if it is existing, and if not then I must first create it with the mandatory fields using **New-MailContact** command and then run the **Set-Contact** command to add the address, phone and non-mandatory fields. Again, this could be treated as a programming logic issue but it is a decent setback from what could be a simple 2 case process.

Problem #3 – The process runs great from my Exchange server, or from my workstation. Since I need to run this from a centralized machine that does a multitude of automated processing, I now need to create a working environment on that server to provide the framework to run this process.

This is where the yak shaving really peaks. Now I have to schedule a formal change to install the Microsoft Exchange Management Shell and associated dependencies on the job server. Once that is done, I need to schedule it to run my script with specific credentials. Now I have a new account to manage. The final issue is that the script is not signed, so I need to configure the server's execution policy to run as Unrestricted which requires a sign off as a potential vulnerability.

The moral of the story is that about 40-50% of the time spent, and about 70% of the total duration (due to wait times for change management and environment testing) is spent on being able to run this process which was unrelated to the actual business logic and programming time. That my friends is what yak shaving is all about.

The good news is that it works, and the better news is that I'm finalizing the script and will be posting it to my TechNet Gallery in the next two weeks so I can hopefully save some of you the shaving time!

That's a little better

PowerShell One-Liners – Take my script, please!

"Doctor, my leg hurts. What can I do?" The doctor says, "Limp!"

Hail Henny Youngman, the king of one-liners! One of the great things about PowerShell is the ability to create simple, powerful one-line scripts. This is not to say that we need to crunch everything down to single line just to be cute, but we have the ability to build effective processes by making use of the pipeline.

With shell scripting, it's often regarded as a number of processes that we pile together with inline code, but quite often we only have some small tasks that we need to run, but they require more than one action in the script.

The long and the short of it (emphasis on short) is that we can do a lot of things with a single line of code by leveraging the pipeline.

Here is an article about piping and the pipeline from Microsoft TechNet: http://technet.microsoft.com/en-us/library/ee176927.aspx

The pipeline allows you to take some input, whether it is a file, or a query and then to perform a

cumulative set of actions.

Let's say I want to read the contents of a text file, and create an output file with only lines that contain the phrase "yourdomain.com". To do this we use this simple one-line script:

> **Get-Content yourinputfile.txt | Where-Object {$_ -match 'yourdomain.com'} | Out-File youroutputfile.txt**

As you can see we are reading the file, applying a where clause to the input and then writing the output to a destination. This is a classic case of a pipeline by taking one object, processing it, and then outputting the processed content.

Let's look at an example where we want to scan Active Directory users with the Quest ActiveRoles Get-QADUser CmdLet and output the Password status to screen but we want to exclude non-expiring passwords:

> **Get-QADUser -SizeLimit 0 | Select-Object samAccountName,mail,PasswordStatus | Where-Object {$_.PasswordStatus -ne "Password never expires" | Write-Host "$_.samAccountName status with email $_.mail has a password status of $_.PasswordStatus"**

This script reads the directory, selects the fields we want and then outputs to screen for us using the Write-Host CmdLet. Simple and effective.

As you can see we can do many tasks where it is **Input => Process => Output** and the Process step can contain many steps within it.

So what's the downside? One-liners aren't always appropriate for a few reasons. Primarily, if we need to loop through input multiple times, or parse the content, run complex calculations. Another reason is that we may be processing very large input which must reside in memory during the process which may require a more carefully designed script to include multiple loops and garbage collection during runtime.

This is an organic and dynamic article where I will collect my favorite one-liners by category over time. This is the starting point and I welcome you to add your own in the comments and I will be sure to add them to the collection.

Last updated on June 24, 2013

Exchange 2010

Query your Exchange servers for databases and file locations

> **Get-MailboxDatabase -Status | select ServerName,Name,DatabaseSize,EdbFilePath, | Export-Csv X:MailboxDatabase.csv**

Report Exchange 2010 Recipient Mailbox Policies (from @JeffHicks)

> **get-mailbox | sort RetentionPolicy | format-table -group RetentionPolicy**

Query your Exchange server queues and show the message count as a point in time snapshot (from @HardcoreITGuy)

Get-exchangeserver | where {$_.isHubTransportServer -eq $true} | get-queue | where {$_.MessageCount -gt 0}

Query your Exchange environment for a list of ActiveSync devices sorted by Device Type and output to CSV

Get-ActiveSyncDevice | select-object DeviceModel,FriendlyName,DeviceOS,UserDisplayname | sort-object DeviceModel | Export-CSV "ActiveSyncDevices.csv"

Windows

Find the installation date of Windows through WMI

([WMI]").ConvertToDateTime((Get-WmiObject Win32_OperatingSystem).InstallDate)

Active Directory

List computer object counts categorized by Operating System (uses Quest ActiveRoles snap in)

Get-QADComputer -sizeLimit 0 | Group-Object -Property OSName,OSServicePack -noelement | Sort-Object -Property Count -Descending | Format-Table -Property Count,Name

Export all XP Computers, with all properties into a CSV format sorted by IP Address

Get-ADComputer -Filter {OperatingSystem -like "Windows XP*"} -Properties * | sort-Object IPv4Address | Export-CSV XPComputers.csv

Export a list of all non-server OS computers into a CSV format where the computer account hasn't contacted AD for over 30 days

get-adcomputer -Filter { OperatingSystem -notlike "Windows Server*" } -Properties PasswordLastSet | Where-Object { (((Get-Date) – $_.PasswordLastSet).Days) -gt 30} | select-object Name, @{Name="Age";Expression={ (((Get-Date) – $_.PasswordLastSet).Days) }} | Export-CSV ComputerListByAge.csv

Much more to come so check back often and I'd love to have your contributions to add and share.

Group Policy – WMI Filters by Operating System

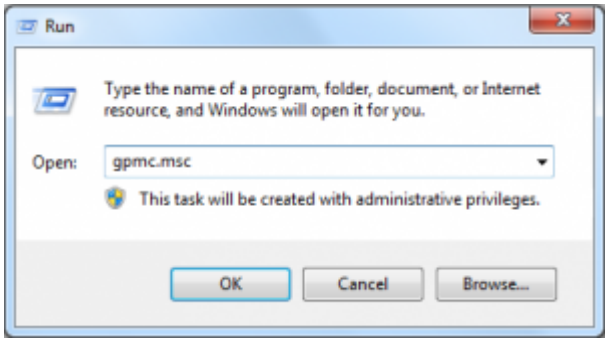
Have you ever wanted to separate your Active Directory Group Policies from each other based on criteria such as target operating system? Well you are in luck! With some very simple WMI filters you can do exactly that.

Using WMI filters is a simple, and flexible way to create specific target criteria for delivering

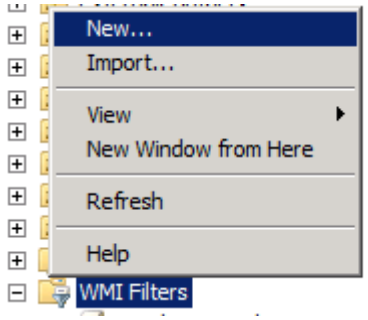
policies. For my situation, I'd like to be able to create 3 specific policies so that I can ensure that there is no contamination of machines with the incorrect configuration.

We could achieve this using OU structure and manually moving around computer objects, but I would much rather be able to let the system do the heavy lifting and guarantee that I do not have any accidental policy delivery, or worse that no policies get deployed at all to the machines.

From a Domain Controller, or from a workstation running the Remote Server Administration Tools (RSAT), launch the Group Policy Management Console **(Start | Administrative Tools | Group Policy Management**) or by running **GPMC.MSC** from the **Run** command.



Expand the Forest and Domain until you will see the WMI Filters folder towards the bottom of the list. Right click the WMI Filters folder and select **New...** to create a new filter.

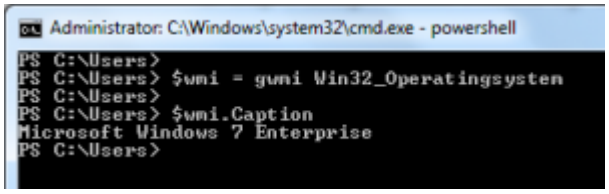


The first policy we will create is one for Windows Server 2008. I do not need to differentiate between editions (Standard, Enterprise, Web) or chip architecture (x86 or x64) so my filter query will be for any version of Windows Server 2008.

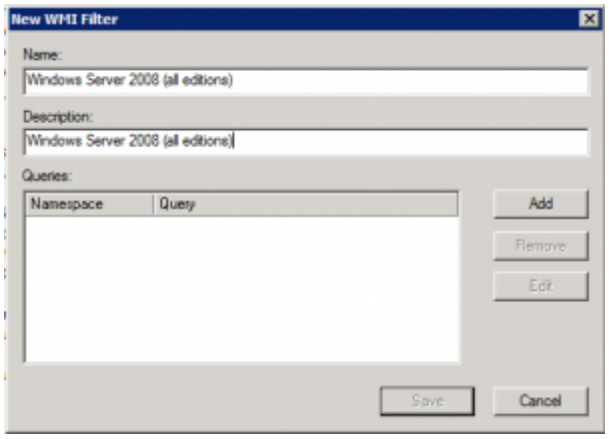
The WMI property we are looking at for this is Caption from the Win32_OperatingSystem. You can look at yours using this simple PowerShell process:

$wmi = gwmi Win32_OperatingSystem

$wmi.Caption



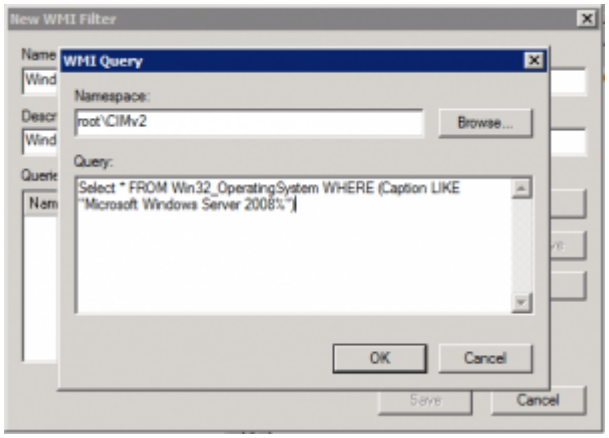
Let's use the name **Windows Server 2008 (all editions)** for the name and description field of the new WMI Query



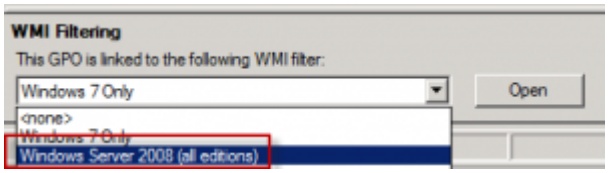
Now click on the Add button which brings up the query window. Leave the Namespace as rootCIMv2 and then under the query section type this:

Select * FROM Win32_OperatingSystem WHERE (Caption LIKE "Microsoft Windows Server 2008%")

The by appending the % to the LIKE query it means that anything found after the 2008 in the Caption will be accepted. You can also use the Version property, but that is a number which is changed by Service Packs and can be more difficult to pinpoint. I'm only in need of knowing the OS type which makes it much easier to use the Caption.



Now that you've saved this new WMI filter, you can go to your Group Policy Object and on the Scope tab at the bottom you use the drop down list to apply your new WMI filter to the policy.



For my other 2 queries, I use the same process but I want to have a Windows XP and a Windows 7 for managing my desktop pools with clearly targeted policies. For my Windows XP filter:

Select * FROM Win32_OperatingSystem WHERE (Caption LIKE "Microsoft Windows XP%")

and for Windows 7:

Select * FROM Win32_OperatingSystem WHERE (Caption LIKE "Microsoft Windows 7%")

It’s just that easy. Now go forth and filter!

Think you are irreplaceable? – Information sharing is a necessity

We have all worked with this person. They are the one who knows the inner workings of all the strange systems, or understands some obscure process that nobody else understands, or perhaps wants to be responsible for. We do this ourselves quite often. When I launch a new system or process I am equally guilty of not always putting the right information into everyone’s hands that need it.

Solution: Share! Sounds overly simple doesn’t it? There are two top reasons why someone is the keeper of information that is not shared and those are:

1. The owner of that information has not had time to document or share that information with their peers
2. The information holder feels a sense of security in being the sole owner

If you are the only person who has the knowledge to make a process or system work I have one piece of news for you: You are replaceable! Regardless of the most important piece of knowledge that you hold with you like a lockbox, you (and I) are not required to keep the machine moving.

This isn’t meant to sound like a grim warning, or that you should not feel like an integral part of your team and your organization, but more as a reminder that although we’ve been told that “knowledge is power” that it is only powerful when shared. If Albert Einstein had developed his solutions for the theory of relativity and then held all of his findings to himself, do you know who would have been famous? Not Eintstein, but the next scientist who worked out the solution.

We can see examples in every aspect of society. Are you familiar with the band Journey? If so, you probably know that distinct, original tenor voice of Steve Perry.

There is no way they can replace me…wait, what?

The seemingly inimitable voice of Steve Perry was the defining sound of Journey throughout their career in the 70s and 80s. Unfortunately, due to a hip replacement which sidelined the front man, it eventually forced the band to make a decision to move forward without him. Enter Arnel Pineda.

Boy Steve Perry really looks different than I remember

I won't go into the story deeply, but it is an interesting one if you choose to search it out. The long and the short of it is that Arnel Pineda can sing with the identical vocal style of Steve Perry. As a result, he is touring with Journey today and has been for many years with great success.

But I digress. The moral that I have captured from this is that to be an important part of any team, whether in work or in your personal life, you should be open and share your knowledge and experience. I have found that with opening the book on my technical knowledge and skills that I have gained much more knowledge capital and have built stronger relationships as a result.