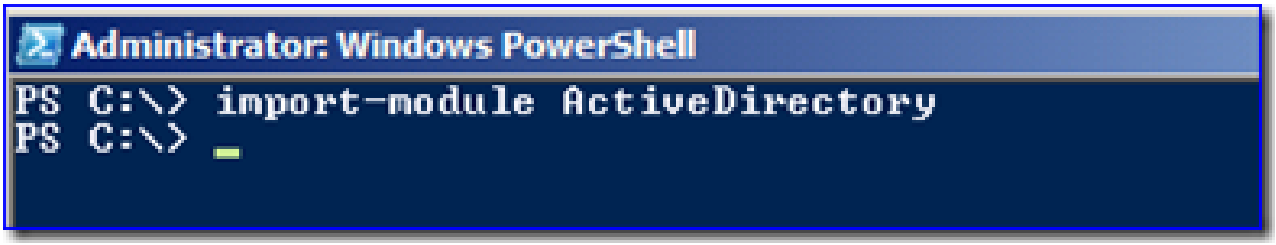
Inventorying Computers with AD PowerShell

techcommunity.microsoft.com/blog/askds/inventorying-computers-with-ad-powershell/397414



Blog Post

First published on TechNet on Feb 04, 2010
Hi, Ned here again. Have you ever had to figure out what operating systems are running in your domain environment so that you can plan for upgrades, service pack updates, or support lifecycle transitions? Did you know that you don't have to connect to any of the computers to find out? It's easier than you might think, and all possible once you start using AD PowerShell in Windows Server 2008 R2 or Windows 7 with RSAT.

Get-ADComputer

The cmdlet of choice for inventorying computers through AD is **Get-ADComputer** . This command automatically searches for computer objects throughout a domain, returning all sorts of info.

As I have written about previously my first step is to fire up PowerShell and import the **ActiveDirectory** module:

Then if I want to see all the details about using this cmdlet, I run:

```
Get-Help Get-ADComputer -Full
```

Getting OS information

Now I want to pull some data from my domain. I start by running the following:

```
Get-ADComputer -Filter * -Property * | Format-Table
Name,OperatingSystem,OperatingSystemServicePack,OperatingSystemVersion -Wrap –Auto
```

Another important note (thanks dloder): I am going for simplicity and introduction here, so the -Filter and -Property switches are not designed for perfect efficiency. As you get comfortable with AD PowerShell, I highly recommend that you start tuning for less data to be returned - the "filter left, format right" model described here by Don Jones.

This command is filtering all computers for all their properties. It then feeds the data (using that pipe symbol) into a formatted table. The only attributes that the table contains are the computer name, operating system description, service pack, and OS version. It also automatically sizes and wraps the data. When run, I see:

It looks like I have some work to do here – one Windows Server 2003 computer needs Service Pack 2 installed ASAP . And I still have a Windows 2000 server that is going to move quickly and replace that server.

Servers

Now I start breaking down the results with filters. I run:

```
Get-ADComputer -Filter {OperatingSystem -Like "Windows Server*"} -Property * |
Format-Table Name,OperatingSystem,OperatingSystemServicePack -Wrap -Auto
```

I have changed my filter to find all the computers that are running “Windows Server *something* ”, using the **–like** filter. And I stopped displaying the OS version data because it was not providing me anything unique (yet!).

Cool, now only servers are listed! But wait… where’d my Windows 2000 server go? Ahhhh… sneaky. We didn’t start calling OS’s “Windows Server” until 2003. Before that it was “Windows 2000 Server”. I need to massage my filter a bit:

```
Get-ADComputer -Filter {OperatingSystem -Like "Windows *Server*"} -Property * |
Format-Table Name,OperatingSystem,OperatingSystemServicePack -Wrap -Auto
```

See the difference? I just added an extra asterisk to surround “Server”.

As you can see, my environment has a variety of Windows server versions running. I’m interested in the ones that are running Windows Server 2008 or Windows Server 2008 R2. And once I have that, I might just want to see the R2 servers – I have an upcoming DFSR clustering project that requires some R2 computers. I run these two sets of commands:

```
Get-ADComputer -Filter {OperatingSystem -Like "Windows Server*2008*"} -Property *
| Format-Table Name,OperatingSystem,OperatingSystemServicePack -Wrap -Auto

Get-ADComputer -Filter {OperatingSystem -Like "Windows Server*r2*"} -Property * |
Format-Table Name,OperatingSystem,OperatingSystemServicePack -Wrap -Auto
```

Starting to make sense? Repetition is key; hopefully you are following along with your own servers.

Workstations

Okeydokey, I think I've got all I need to know about servers – now what about all those workstations? I will simply switch from - **Like** to **-Notlike** with my previous server query:

```
Get-ADComputer -Filter {OperatingSystem -NotLike "*server*"} -Property * | Format-Table Name,OperatingSystem,OperatingSystemServicePack -Wrap -Auto
```

And blammo:

OS Family

By now these filters should be making more sense and PowerShell is looking less scary. Let's say I want to filter by the "family" of operating system. This can be useful when trying to identify computers that started having a special capability in one OS release and all subsequent releases, and where I don't care about it being server or workstation. An example of that would be **BitLocker** – it only works on Windows Vista, Windows Server 2008, and later. I run:

```
Get-ADComputer -Filter {OperatingSystemVersion -ge "6"} -Property * | Format-Table Name,OperatingSystem,OperatingSystemVersion -Wrap -Auto
```

See the change? I am now filtering on operating system version, to be equal to or greater than 6. This means that any computers that have a kernel version of 6 (Vista and 2008) or higher will be returned:

If I just wanted my Windows Server 2008 R2 and Windows 7 family of computers, I can change my filter slightly:

```
Get-ADComputer -Filter {OperatingSystemVersion -ge "6.1"} -Property * | Format-Table Name,OperatingSystem,OperatingSystemVersion -Wrap -Auto
```

Getting it all into a file

So what we've done 'til now was just use PowerShell to send goo out to the screen and stare. In all but the smallest domains, though, this will soon get unreadable. I need a way to send all this out to a text file for easier sorting, filtering, and analysis.

This is where **Export-CSV** comes in. With the chaining of an additional pipeline I can find all the computers, select the attributes I find valuable for them, then send them into a comma-separated text file that is even able to read the weirdo UTF-8 trademark characters that lawyers sometimes make us put in AD.

Hey, what do you call a million lawyers at the bottom of the ocean? A good start! Why don't sharks eat lawyers? Professional courtesy! What do have when a lawyer is buried up to his neck in sand? Not enough sand! Haw haw… anyway:

```
Get-ADComputer -Filter * -Property * | Select-Object
Name,OperatingSystem,OperatingSystemServicePack,OperatingSystemVersion | Export-
CSV AllWindows.csv -NoTypeInformation -Encoding UTF8
```

Then I just crack open the **AllWindows.CSV** file in Excel and:

What about the whole forest?

You may be tempted to take some of the commands above and tack on the necessary arguments to search the entire forest. This means adding:

```
-searchbase “” –server  <domain FQDN>  :3268
```

That way you wouldn’t have to connect to a DC in every domain for the info – instead you’d just ask a single GC. Unfortunately, *this won’t work* ; none of the operating system attributes are replicated by global catalog servers. Oh well, that’s not PowerShell’s fault. All the data must be pulled from domains individually, but that can be automated – I leave that to you as a learning exercise.

Conclusion

The point I made above about support lifecycle is no joke: 2010 is a very important year for a lot of Windows products’ support:

- Windows XP ***SP2*** support ends July 13, 2010 (mainstream support for the whole OS ended in 2009 and you must be running SP3 after July 13)
- Windows 2000 Professional ***extended*** support ends July 13, 2010
- Windows 2000 Server ***extended*** support ends July 13, 2010
- Windows Server 2003 and Windows Server 2003 R2 ***mainstream*** support ends July 13, 2010 (SP1 support ended in 2009, you must be running SP2 now)
- Windows Vista ***RTM*** (no service pack) support ends April 13, 2010 (You are running Vista with no service pack? Really?)
- For more info on what “support” really means, head over to the Lifecycle page.

Hopefully these simple PowerShell commands make hunting down computers a bit easier for you.

Until next time.

- Ned “bird dog” Pyle
Updated Jun 20, 2023