Becoming a PowerShell ConfigMgr Ninja is CIMple!

Merlijn Van Waeyenberghe

Thank you sponsors!

Gold Sponsor

Patch My PC
PATCH MANAGEMENT MADE EASY

Sponsors

DIGICOMP

RealStuff
Informatik AG

baseVISION
SECURE & MODERN WORKPLACE

Agenda

Becoming a PowerShell ConfigMgr Ninja is CIMple!

Key takeaways:

- **Understand WMI vs CIM**
- **Make your code future-proof**
- **Improve code performance**

About Merlijn

Principal Consultant at OB·V·US IT'S OBVIOUS

Focus

Microsoft 365 Workplace Management

Automation

Belgian PowerShell User Group @BEPUG

From

My Blog

https://www.obvus.be/blog/

Certifications

M365, Power Platform, MCSA, CPL(H)

Hobbies

Travel

Aviation

Riding my bike

Virtual Reality

Contact

@MerlinFromBE

merlinfrombelgium

PowerShell and ConfigMgr

- In the context of MEM Configuration Manager, PowerShell is used for
  - Configuration Manager PowerShell Module
  - Application Deployment Types and Detection Methods
  - Configuration Item Detection and Remediation
  - Task Sequence script steps
  - CMPivot queries on clients
  - Scripts to run on ConfigMgr clients
  - Pre- and Post-action scripts for Orchestration Groups

- ConfigMgr requires at least PowerShell version 3
  - CMPivot requires v4
  - Cmdlets and functions used in scripts may require higher version – check compatibility using PSScriptAnalyzer
    https://devblogs.microsoft.com/powershell/using-psscriptanalyzer-to-check-powershell-version-compatibility/
  - Recommended version on Windows: PowerShell 5.1
  - PowerShell 7 is NOT YET supported by ConfigMgr Current Branch – but support is added in TP 2004

WMI vs CIM

CIM = WMI = CIM *

* Source: https://devblogs.microsoft.com/scripting/should-i-use-cim-or-wmi-with-windows-powershell/

- WMI (Windows Management Instrumentation) is Microsoft's implementation of the open-source CIM (Common Information Model)
- CIM cmdlets introduced in PS 3.0
- WMI cmdlets deprecated since PS 6.0
  - PS 7 is built on .NET Core, which does not support WMI
  - WMI can still be queried
    - WMI is a technology unrelated to PowerShell
    - Just use the CIM cmdlets with the same query, for example:

```
Get-WmiObject Win32_OperatingSystem
=
Get-CimInstance Win32_OperatingSystem
```

WMI	CIM
Get-WmiObject	Get-CimInstance
Set-WmiObject	Set-CimInstance
Invoke-WmiMethod	Invoke-CimMethod
Remove-WmiObject	Remove-CimInstance
Register-WmiEvent	Register-CimIndicationEvent

configmgr.ch

WMI vs CIM

Tips & Tricks

- CIM objects are snapshots of the server-side WMI object
- $CimObject.Get() and other methods cannot be invoked like they could on WMI objects
- Invoke-CimMethod does not require a strictly ordered list of arguments like the WMI method does
- Create new instances using
  - New-CimInstance -ClassName <something> -Property @{Property1 = ‘string’; Property2 = 1; Property3 = $anotherCimClass}
- OR create a class object first and edit properties on the fly
  - $class = Get-CimClass <something>
  - $newInstance = New-CimInstance -ClientOnly -CimClass $class
  - $ newInstance.Property1 = ‘string’
  - Set-CimInstance -InputObject $newInstance

configmgr.ch

New cmdlets

- Classes and instances

Get-CimClass
Get-CimAssociatedInstance
New-CimInstance

- Benefits
  - Use <TAB> to navigate through WMI
  - Find classes that have property ‘x’ or method ‘y’
  - No longer needs ordered properties for methods and new WMI objects

- Sessions

Get-CimSession
New-CimSession
New-CimSessionOption
Remove-CimSession

- Benefits
  - Remote session is a client-side object
    = less overhead, better performance
  - WSMAN (WinRM) instead of DCOM
  - Can still fall back on DCOM if target is PS 2.0

Demo: CIM cmdlets

Demo: performance

configmgr.ch

Comparison study

Command	Session	Module	Server-side footprint	Performance index
~~Get-WmiObject -Computername -Class SMS_Collection~~	local		👍	∧
Get-CimInstance -Computername -Class SMS_Collection	local		👍	∧
~~Invoke-Command -Session -Command {Get-WmiObject -Class SMS_Collection}~~	remote		👎	—
Invoke-Command -Session -Command {Get-CimInstance -Class SMS_Collection}	remote		👎	—
Get-CimInstance -CIMSession -Class SMS_Collection	remote		👍	∧
Get-CMDeviceCollection	local	Configuration Manager	👍	∨
Invoke-Command -Session -Command {Get-CMDeviceCollection}	remote	Configuration Manager	👎	∨
Get-CCMCollection	remote	CCM	👍	∧

Convert your code

- First, review where and how your code runs
  - Do multiple scripts run on one central location? > consider resource impact and conflicting schedules
  - Does your script run on the CM server or remotely? > make sure all automation runs remotely!
  - What user does it run as? > dedicated user accounts with least privilege is preferred

- Then establish your options

Current solution	My favourite option	Alternative options
ConfigurationManager Module	CCM Module	CIM session
PSSession		
Invoke-Command {Get-WmiObject}	CIM session	CCM Module
Get-WmiObject -ComputerName		

Demo: code conversion

Helpful resources and tools

- https://devblogs.microsoft.com/scripting/should-i-use-cim-or-wmi-with-windows-powershell/
- https://devblogs.microsoft.com/scripting/comparing-powershell-pssessions-and-cim-sessions/
- https://docs.microsoft.com/en-us/powershell/scripting/learn/ps101/07-working-with-wmi?view=powershell-7
- https://maikkoster.com/powershell-cim-cmdlets-working-with-lazy-properties/
- https://msdnshared.blob.core.windows.net/media/MSDNBlogsFS/prod.evol.blogs.msdn.com/CommunityServer.Components.PostAttachments/00/10/36/34/70/WMI%20CheatSheet%20for%20PS.pdf
- https://github.com/saladproblems/CCM-Core

Thank you sponsors!

Gold Sponsor

Patch My PC
PATCH MANAGEMENT MADE EASY

Sponsors

DIGICOMP

RealStuff
Informatik AG

baseVISION
SECURE & MODERN WORKPLACE

Thank You