

# Using CIM Commands

---



**Liam Cleary**

CEO / MICROSOFT MVP / MICROSOFT CERTIFIED TRAINER

@shareplicity [www.shareplicity.com](http://www.shareplicity.com) | @helloitsliam [www.helloitsliam.com](http://www.helloitsliam.com)



# Overview



What is CIM?

WMI versus CIM

Executing commands using CIM



# What is CIM?

---



# CIM

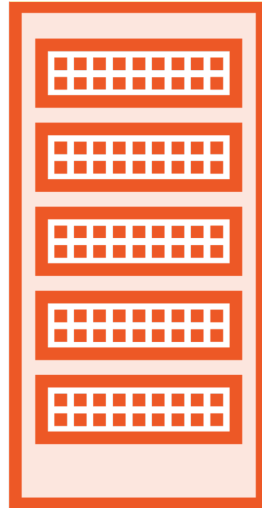
Stands for Common Information Model. It is a set of standards that describe how information is structured and represented within a system



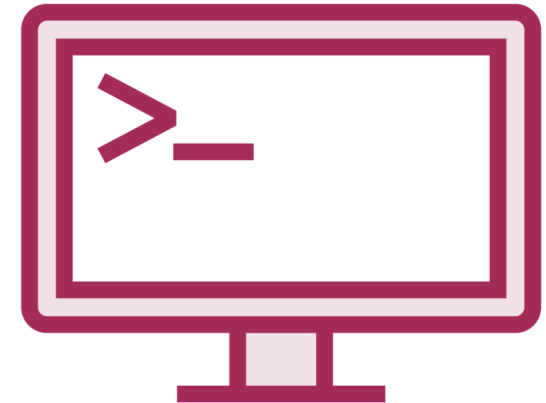
# What is CIM?



Introduced in  
PowerShell 3.0



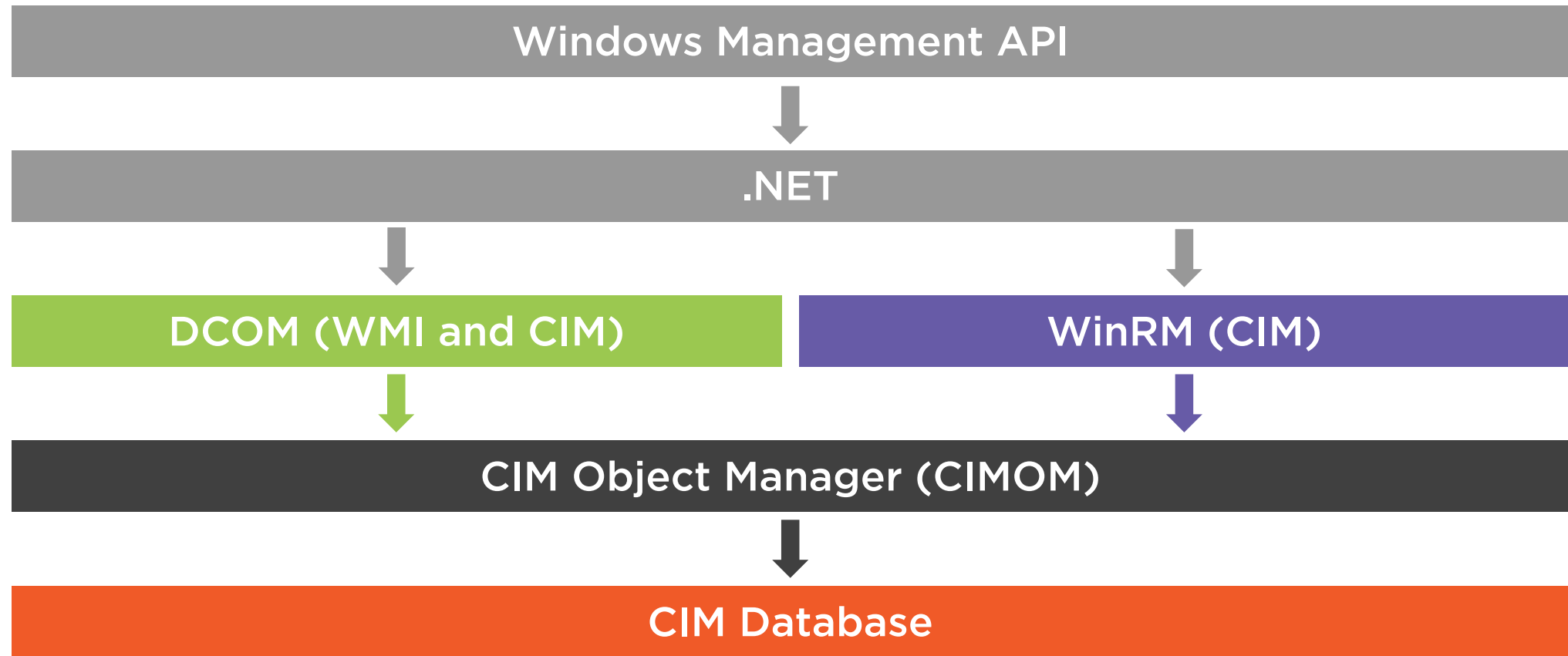
Available from  
Windows Server 2012  
and Windows 8



Replacement for  
existing WMI  
commands



# CIM Architecture



# WMI Versus CIM

---



# WMI Versus CIM

## Deprecated Commands

Not included with PowerShell Core

Uses DCOM for communication

Example: *Get-WmiObject*

## New Commands

Included within PowerShell Core

Uses the WSMAN protocol  
for communication

Example: *Get-CimInstance*





# Comparing Commands

**# Using WMI to Retrieve "Operating System" Details**

```
Get-WmiObject -Class Win32_OperatingSystem
```

**# Using CIM to Retrieve "Operating System" Details**

```
Get-CimInstance -ClassName Win32_OperatingSystem
```



# How to Find CIM Classes and Instances

**# List all CIM Classes**

```
Get-CimClass
```

**# List all Win32 Disk Related Classes**

```
Get-CimClass Win32*Disk*
```

**# List CIM Class with specific Method Name**

```
Get-CimClass -ClassName Win32* -MethodName Term*
```

**# Get Instances for specific Class**

```
Get-CimInstance -ClassName Win32_Process
```

**# Get CIM Namespaces**

```
Get-CimInstance -Namespace root -ClassName __Namespace
```



# Using the CIM Explorer

The screenshot shows the CIM Explorer application window. At the top, there are two tabs: 'CIM Explorer' (active) and 'Commands'. Below the tabs, the main pane displays a tree view. The first level is 'Connect to liamcleary896b (DCOM)'. Expanding this shows 'Namespace - root'. Under 'root', there are several namespaces: 'subscription', 'DEFAULT', 'CIMV2', 'msdtc', and 'Cli'. The 'Cli' namespace is selected, and its contents are displayed in the lower pane. The lower pane shows 'Namespace - ROOT/Cli' and 'Class'. Below this, a list of classes is shown: 'MSFT\_CliConnection', 'MSFT\_CliFormat', 'MSFT\_CiParam', 'MSFT\_CliProperty', and 'MSFT\_CliQualifier'. Four arrows point to specific elements in the interface: one to the 'Connect to liamcleary896b (DCOM)' entry, one to the 'root' namespace, one to the 'Cli' namespace, and one to the list of classes.

**Enumerate Namespaces**

**Enumerate Classes**

**Connect to Windows Machine**

**Click into Namespace to view Classes**



# Demo



## Enumerate Namespaces and Classes

### Using the CIM Class Explorer in PowerShell ISE

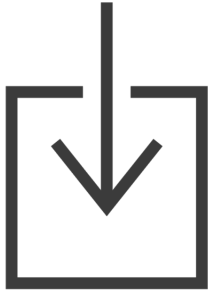


# Executing commands using CIM

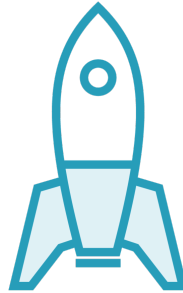
---



# CIM Command Verbs



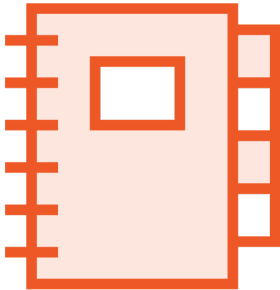
GET



INVOKE



NEW



REGISTER



REMOVE



SET

# Common CIM Commands



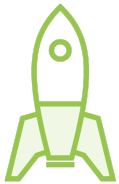
## **Get-CimInstance**

Gets the CIM instances of a class from a CIM server



## **New-CimSession**

Creates a CIM session.



## **Invoke-CimMethod**

Invokes a method of a CIM class



# Get-CimInstance

**# Query all the CIM Instances starting with the letter P in class Win32\_Process**

```
Get-CimInstance -Query "SELECT * from Win32_Process WHERE name LIKE 'P%'"
```

```
Get-CimInstance -ClassName Win32_Process -Filter "Name like 'P%'"
```

**# Retrieve specific properties for CIM Instance**

```
Get-CimInstance -Class Win32_Process -Property Name, KernelModeTime
```

```
Get-CimInstance -Class Win32_Process -Property Name, KernelModeTime | Format-Table
```





# New-CimSession

**# Create a CIM Session with default options**

```
New-CimSession
```

**# Create a CIM Session with a friendly name**

```
New-CimSession -ComputerName localhost -Name Trainer
```

**# Pass credentials to a CIM Session**

```
$creds = Get-Credential
```

```
New-CimSession `
    -ComputerName localhost `
    -Credential $creds `
    -Authentication Negotiate
```



# Invoke-CimMethod

**# Terminate a running "Console" process**

```
Invoke-CimMethod `
    -Query 'Select * From Win32_Process Where Name Like "cmd%"' `
    -MethodName "Terminate"
```

**# Create a running "Console" process with "Arguments"**

```
Invoke-CimMethod `
    -ClassName Win32_Process `
    -MethodName "Create" `
    -Arguments @{
        CommandLine = 'cmd.exe'; CurrentDirectory = "C:\windows\system32"
    }
```



# Demo



## Execute Commands using CIM



# Summary



Reviewed what CIM is as well as how it works

Executed commands using CIM locally



# Up Next: Managing Computers using CIM Commands

---

