

Managing Computers with PowerShell and CIM

UNDERSTANDING WMI IN POWERSHELL



Liam Cleary

CEO / MICROSOFT MVP / MICROSOFT CERTIFIED TRAINER

@shareplicity www.shareplicity.com | @helloitsliam www.helloitsliam.com



Overview



What is WMI?

Executing commands using WMI

Managing Computers using WMI



What is WMI?



WMI

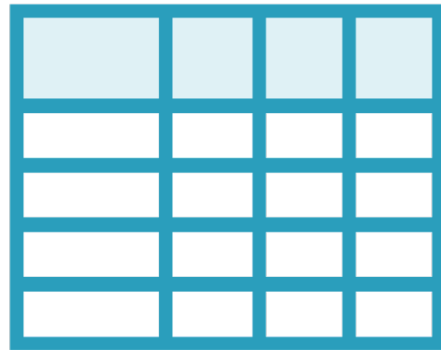
Stands for Windows Management Instrumentation. It is Microsoft's implementation of Web-Based Enterprise Management (WBEM) allowing access to data



What can WMI be used for?



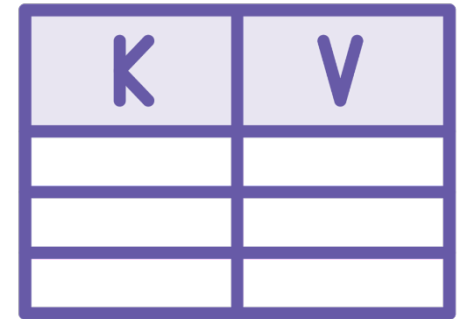
**Set Security
Settings**



**Collect
Information**



**Set and Change
User
Permissions**



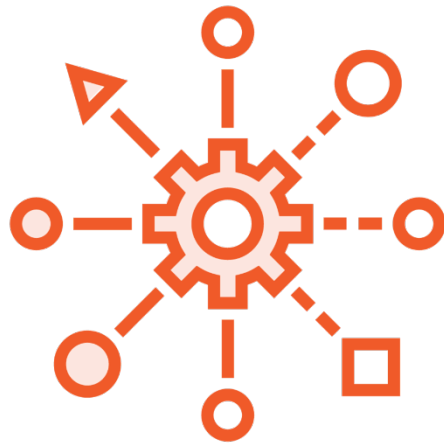
**Configure
System
Properties**



What can WMI be used for?



Schedule
Processes to
Run



Manage Code
Execution



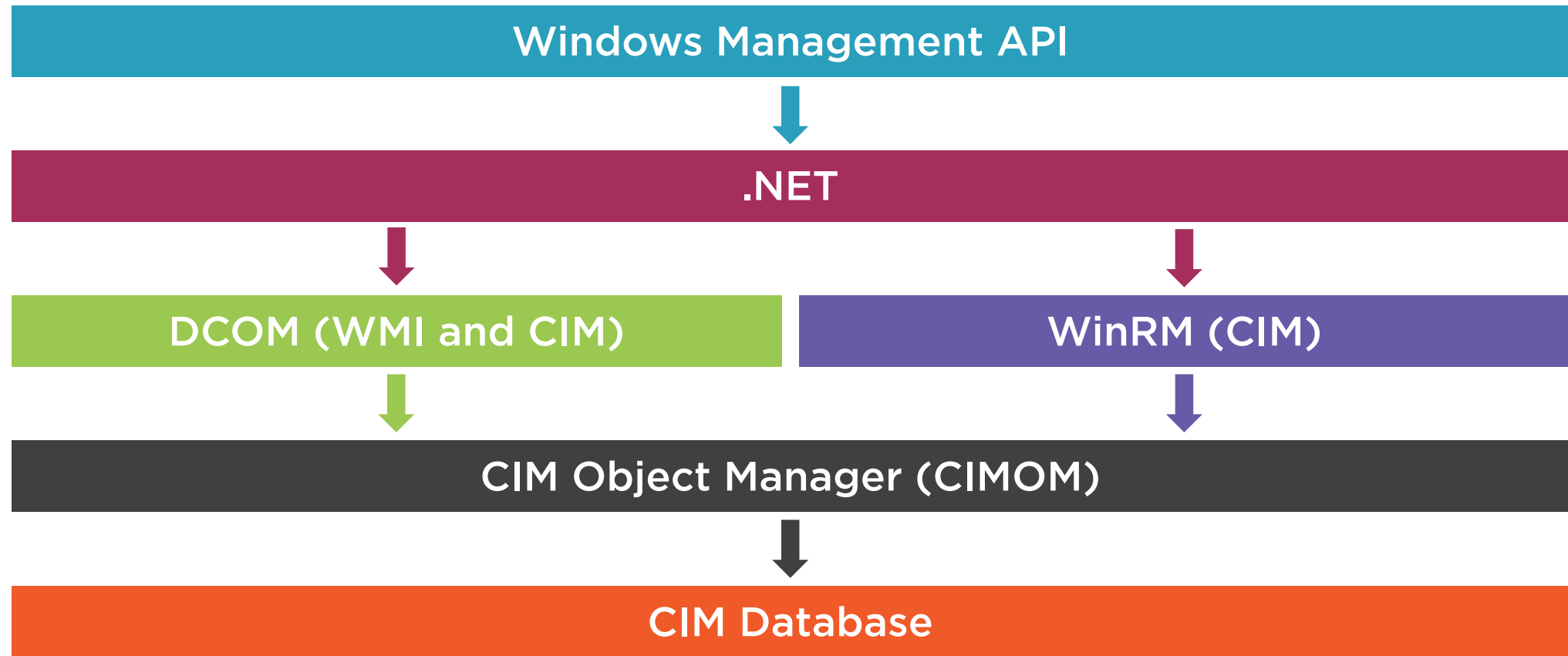
Manage Error
Logging



Manage Drives



WMI Architecture



WMI Components



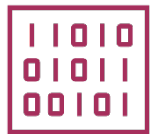
WMI Service is the implementation in Windows of the WMI system



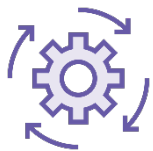
Managed Objects are any logical or physical component or service that can be managed via WMI



WMI Providers are objects that monitor events and data from a specific object



Classes are used by WMI providers to pass data to WMI services



Methods are attached to classes and allow actions to be performed based on data included in them



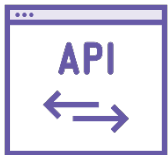
WMI Components



WMI repository is a database that stores all the static data that is related to WMI



CIM Object Manager is a system that sits in between a management application and WMI providers



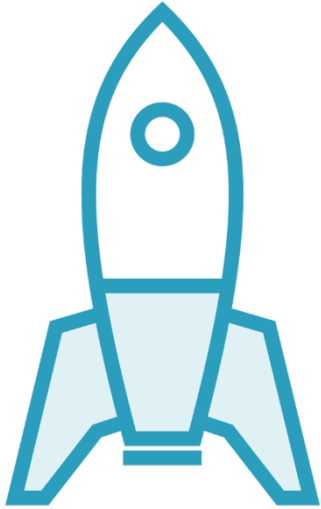
WMI API provides a way for applications to access the WMI infrastructure



WMI Consumer is the entity that sends queries to objects via the Object Manager



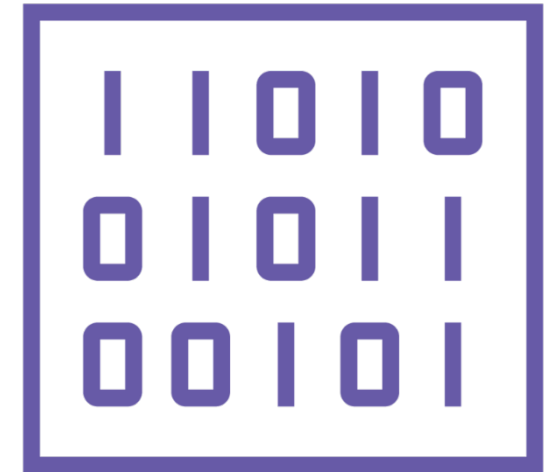
WMI Structure



Namespaces are in a file system structure that organizes the objects into functions



Class Instances are the objects stored within the Namespaces



Operating System and Application specific data is exposed via the Class Instances



Demo



Review WMI Namespaces



Executing commands using WMI



Core Parameters



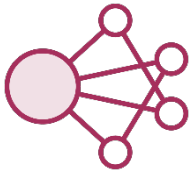
ComputerName

Execute the specified command on a remote computer



List

Used to get information about the WMI classes that are available in a specified namespace



Query

Executes a WMI query language (WQL) statement



Command Structure

Command

Get-WmiObject

Class

Win32_Processor

Remote
Machine

Trainer

Query

Select * From
Win32_Service
Where
Name='WinRM'

Get-WmiObject `

-Class Win32_Process `

-ComputerName Trainer `

-Query "Select * From Win32_Service Where Name='WinRM'"



Retrieving Processes on Computers

Retrieve processes on the local computer

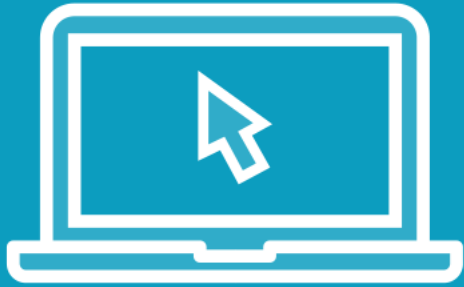
```
Get-WmiObject -Class Win32_Process
```

Retrieve processes on a remote computer

```
Get-WmiObject -Class Win32_Process -ComputerName workstation
```



Demo



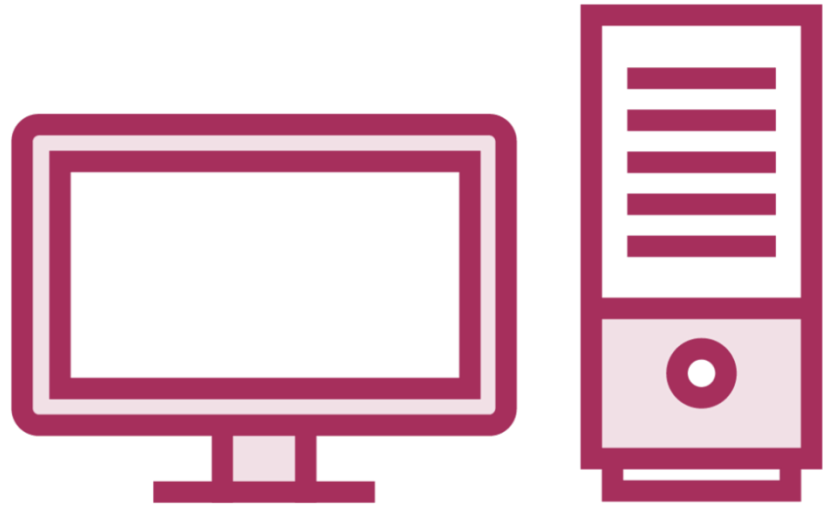
Execute Basic WMI Commands



Managing Computers using WMI



Managing Computers using WMI



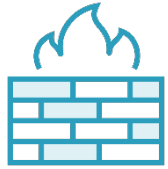
Manage Local Machine



Manage Remote Machine(s)



Prerequisites for Managing Computers



Windows Firewall Settings

Enable WMI traffic through the Windows Firewall



User Account Control Settings

User "Run as Administrator" when running PowerShell



DCOM Settings

Explicitly grant remote DCOM access, activation, and launch rights to the account used for connecting



CIMOM Settings

Set the "*AllowAnonymousCallback*" registry key, for machines not on the same domain and untrusted

Retrieving List of Classes and Namespaces

Retrieve all WMI Classes (default Namespace)

```
Get-WMIObject -List
```

Retrieve all WMI Classes where the Name contains "Win32_"

```
Get-WMIObject -List | Where-Object { $_.name -match "^Win32_" }
```

Retrieve list of all WMI Namespaces

```
Get-WmiObject -Namespace Root -Class __Namespace
```

```
Get-WmiObject -Query "Select * From __Namespace" -Namespace Root
```



Retrieving List of Classes and Namespaces

Retrieve all WMI Classes for a specific Namespace

```
Get-WMIObject -Namespace ROOT\SecurityCenter2 -List
```

Retrieve all "Antivirus Product" details

```
Get-WmiObject -Namespace ROOT\SecurityCenter2 -Class AntivirusProduct
```



Connect to a Remote Computer

Connect to Local Computer and Retrieve Operating System Details

```
Get-WmiObject Win32_OperatingSystem -ComputerName localhost
```

Connect to Remote Computer and Retrieve Operating System Details

```
$computer = "Trainer"
```

```
Get-WmiObject Win32_OperatingSystem -ComputerName $computer
```

Connect to Remote Computer using Credentials, and Retrieve Running Processes

```
$computer = "Trainer"
```

```
Get-WmiObject -Namespace "root\cimv2" `
    -Class Win32_Process `
    -Impersonation 3 `
    -Credential DOMAIN\account `
    -ComputerName $computer
```



Retrieve Computer Information

Retrieve all properties for Operating System

```
$computer = "Trainer"
```

```
Get-WmiObject -ComputerName $computer `
    -Class Win32_OperatingSystem | Select-Object -Property *
```

Select specific Operating System property

```
$computer = "Trainer"
```

```
$object = (Get-WmiObject -ComputerName $computer -Class Win32_OperatingSystem)
```

```
$object | (Select-Object -Property *).Caption
```

```
$object | (Select-Object -Property *).Manufacturer
```



Managing Services on Remote Computers

Retrieve "Windows Update" Service Details

```
$computer = "Trainer"  
$service = Get-WmiObject -ComputerName $computer -Class Win32_Service `   
            -Filter "Name='wuauserv' "
```

Check Service has a Start / Stop Method

```
$service | Get-Member -Type Method
```

Start and Stop the "Windows Update" Service

```
$service.stopservice()  
$service.startservice()
```



Demo



Connect to Remote Computers

- Retrieve Information
- Managing Services



Summary



Reviewed what WMI is as well as how it works

Executed commands using WMI, locally and on remote computers



Up Next:
Using CIM Commands

