PowerShell WMI Tutorial!

1 INFORMATION ABOUT THE TUTORIAL

Hi! My name is Sigurd Skauvik and i wrote this tutorial so that you can learn about the power of powershell. Expecially the WMI. Notice that the code in this tutorial is marked with red. Most of this tutorial is based on examples and I will explain the code breifly afterwards. WMI stands for Windows Management Instruments. Its kind of like a pack of namespaces (I will explain in more detail later) and tools for managing the computer. The WMI is a very large package, only in \root\CIMV2 On my machine I have 557 different classes in this namespace! I will recommend downloading a tool called Scriptomatic by Ed Wilson, the great founder of PowerShell step by step book witch is a good book I will recommend for starters.

```
Introduction to WMI:
```

There are 5 cmdlets that operates with WMI classes. They are:
- Get-WmiObject (this is the one we will be using most of the time)
- Invoke-WmiMethod
- Register-WmiEvent
- Remove-WmiObject (this sounds potentially dangerous, but I don't know)
-Set-WmiInstance

Get-WmiObject uses namespace, computername and class. You can filter wmi objects using WQL to use queries you need the -query switch after get-wmiobject. You can also filter using the pipeline and the WHERE clause. Here is an example:

```
Get-WMIObject -Class Win32_LogicalDisk | Where-Object {
$_.DriveType -eq 3
```

Now, let me explain what this does. First it gets the Win32_LogicalDisk class, pipe it through the pipeline and say, we want the disktypes with the value equal 3. Now, the value 3 means logical disks. So this will output information about the logical disks on the machine. This is NOT like using a where clause for WQL (like a query) but instead we have used Where-Object to sort it out. The $_. is the name of the object (win32_logicaldisk).

Here is another example.

```
Get-WMIObject -Query "SELECT * FROM Win32_LogicalDisk WHERE DriveType=3"
```

Now let me explain what this does. Here we have used the –Query parameter to select the harddrive from the class WHERE the drivetype is logical disks. This is close to WQL query scripting. If you have used SQL you would recognize the SELECT * FROM statement. Its exactly the same.

NOTE: -Filter is a variation of -Query. In fact, the value of -Filter represents the value of a WHERE clause when using -Query parameter. When using -Filter, Get-WMIObejct cmdlet internally builds the WMI query as required.

Here is an example where we use WMISEARCHER. I will not go very much into that, I just want you to know that its there.. Windows PowerShell provides shortcuts to allow direct access to .NET namespaces. So, here is an example of WMISEARCHER

```
$svcs = [WMISEARCHER]"SELECT * FROM Win32_Service WHERE State='Running'"
$svcs.Get()
```

Let me explain this a bit. we are using the [WMISEARCHER] shortcut to get a list of all services in “running” state. By default, the scope of the query is set to root\cimv2 namespace.If you want a different scope than root\cimv2 namespace you can do this so:

```
$objSearch = [WMISEARCHER]"SELECT * FROM MSPower_DeviceEnable"
$objSearch.Scope = "root\WMI"
$objSearch.Get()
```

WMI data queries are the simplest form of querying for WMI data. WMI data queries are used to retrieve class instances and associations. So, here is an example

```
$query = "SELECT * FROM Win32_Service WHERE Name='AudioSrv'"
Get-WMIObject -Query $query
```

This is a simple snipplet of code. In the query we have included a WHERE clause to filter out the AudioSrv service from Win32_Service. Notice that classes in WMI use service, not services. And process, not processes. Good to keep in mind.

in WQL, ‘%’ is the wild card character when using LIKE keyword. Here is an example using WQL (Witch is kind of like SQL queries in powershell. So here is an example:

```
$query = "SELECT * FROM Win32_Service WHERE Name LIKE '%Audio%'"
Get-WMIObject -Query $query
```

You probably get the point here. Instead of filter we use LIKE and this will print out anything with Audio in the service name. Filter will NOT be used later in this tutorial, we use -query ONLY :)

In case you need to filter out everything starting from a to z, this is how you do this:

```
$query = "SELECT * FROM Win32_Service WHERE Name LIKE '[a=f]%'"
Get-WMIObject -Query $query
```

I have written down some other common statements in WQL queries:

FROM	Specifies the class that contains the properties listed in a SELECT statement. WMI supports data queries from only one class at a time
AND	Combines two Boolean expressions, and returns TRUE when both expressions are TRUE.
IS	(COMPARING NULL AND NOT NULL The syntax for this statement is the following: IS [NOT] NULL (where NOT is optional)
LIKE	Operator that determines whether or not a given character string matches a specified pattern.
OR	Combines two conditions. When more than one logical operator is used in a statement, the OR operators are evaluated after the AND operators
SELECT	Specifies the properties that are used in a query
TRUE	Boolean operator that evaluates to -1 (minus one).
FALSE	Boolean operator that evaluates to 0 (zero).

Lets look at the WQL operators while we are looking at tables. Here is the WQL operators:

=	Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
!= or <>	Not equal to

I have another example for you using an operator, the AND. I will just write it here and the comment on it afterwards:

```
$query = "SELECT * FROM Win32_Service WHERE State='Running' AND StartMode='Manual'"
Get-WMIObject -Query $query
```

Here we have the same query as last one BUT with one little addition. The AND statement. This will print out every service that are both Running AND have StartMode "Manual"
So, this must be true for the WMIObject to print the service. Simple! Use this example and experiment with the OR operator.

Now, here is an example using the NOT operator:

```
$query = "SELECT * FROM Win32_Service WHERE NOT (State='Running')"
Get-WMIObject -Query $query
```

This will print out services that are NOT running.

SELECT queries always return a collection of instances of a WMI class whereas “ASSOCIATORS OF” Returns a collection of WMI objects that belong to different WMI classes or associated WMI classes.
Example for Associaters of:

```
$query = "ASSOCIATORS OF {Win32_Service.Name='NetLogon'}"
Get-WMIObject -Query $query
```

That’s all for now folks! Thank you for reading my tutorial and keep the spirit alive!

Sigurd Skauvik