

TEC US 2011: PowerShell Deep Dive: Richard Siddaway – WMI: Gems and Gotchas

These are the scripts Richard was using. You can also find his:

- Slides and video here: <http://dmitrysoznikov.wordpress.com/2011/09/14/video-richard-siddaway-wmi-gems-and-gotchas/>

Gems

remove-wmiobject-demo.ps1

```
1..2| foreach {Start-Process notepad}
Get-WmiObject Win32_Process -Filter "Name = 'notepad.exe'" |
select Name, Handle, ProcessId

## delete
Get-WmiObject Win32_Process -Filter "Name = 'notepad.exe' " |
Remove-WmiObject
```

```
1..2| foreach {Start-Process notepad}
Get-WmiObject Win32_Process -Filter "Name = 'notepad.exe'" |
select Name, Handle, ProcessId
```

```
## delete
## check processid !!!!!
Get-WmiObject Win32_Process `
-Filter "Name='notepad.exe' AND ProcessId=3972" |
Remove-WmiObject
```

wmiASSOCIATORS.ps1

```
Get-Process p*

## could use where but try this
Get-WmiObject Win32_Process -Filter "Name LIKE 'p*'" | ft Name, ProcessId -a

Get-WmiObject Win32_Process -Filter "Name LIKE 'p%'" | ft Name, ProcessId -a

Get-WmiObject Win32_Process -Filter "Name LIKE 'power__.%'" | ft Name, ProcessId -a
```

wmiConvertDate.ps1

```
Get-WmiObject Win32_OperatingSystem | ft Lastbootuptime

Get-WmiObject Win32_OperatingSystem | ft @{Name="BootTime";
Expression={$_.ConvertToDateTime($_.Lastbootuptime)}}
```

wmiEvent.ps1

```
function eventhandler {
param ($e)
    $e | fl * | Out-Host
    $e.SourceEventArgs.NewEvent | fl * | Out-Host
}
```

```
if (Get-EventSubscriber | where {$_.SourceIdentifier -eq "delfile"}) { Unregister-Event
-SourceIdentifier "delfile" }
```

```
$dquery = "SELECT * FROM __InstanceDeletionEvent WITHIN 5 WHERE TargetInstance ISA
'CIM_DirectoryContainsFile' AND TargetInstance.GroupComponent =
'Win32_Directory.Name=\"C:\\\\Tesz2\"'"
```

```
$action = {eventhandler $($event)}
```

```
Register-WmiEvent -Query $dquery -SourceIdentifier "delfile" -Action $action
```

wmiSystem.ps1

```
Get-WmiObject -List __*
```

wmiWildcard.ps1

```
Get-Process p*
```

```
## could use where but try this
```

```
Get-WmiObject Win32_Process -Filter "Name LIKE 'p*'" | ft Name, ProcessId -a
```

```
Get-WmiObject Win32_Process -Filter "Name LIKE 'p%'" | ft Name, ProcessId -a
```

```
Get-WmiObject Win32_Process -Filter "Name LIKE 'power__.*'" | ft Name, ProcessId -a
```

Gotchas

wmiDefaultFormat.ps1

```
## also note the introduction of logical processors
```

```
## and how it doesn't report properly in 2003/XP
```

```
Get-WmiObject Win32_ComputerSystem
```

```
Get-WmiObject Win32_ComputerSystem | fl *
```

wmidocument.ps1

```
#####
```

```
## create your own documentation
```

```
function get-wmidoc {
```

```
param (
```

```
    [string]$class
```

```
)
```

```
$data = Get-WmiObject -List $class -Amended
```

```
"`nDescription"
```

```
($data.Qualifiers | where {$_.Name -eq "Description"}).Value
```

```
"`nMethods"
```

```
$data | select -ExpandProperty methods
```

```
"`nProperties"
```

```
$data | select -ExpandProperty properties
```

```
"`nQualifiers"
```

```
$data | select -ExpandProperty qualifiers
```

```
}
```

```
get-wmidoc Win32_process
```

wmikey.ps1

```
## 'root\cimv2' namespace
Get-WmiObject -List | Format-Wide -Column 2

## namespaces
function get-namespace {
param ([string]$name)
    Get-WmiObject -Namespace $name -Class "__NAMESPACE" |
    foreach {
        "$name\" + $_.Name
        get-namespace "$name\" + $_.Name
    }
}
"root"
get-namespace "root"
#####

Start-Process notepad

[wmi]"Win32_Process.Name='notepad.exe'" |
select Name, Handle, ProcessId

#####
## discover the key
function get-key {
[CmdletBinding()]
param (
    [string]
    [ValidateNotNullOrEmpty()]
    $class
)
    $t = [WMIClass]$class
    $t.properties |
    select @{Name="PName";Expression={$_.name}} -ExpandProperty Qualifiers |
    where {$_.Name -eq "key"} |
    foreach {"The key for the $class class is $($_.Pname)"}
}
get-key Win32_Process

Get-WmiObject Win32_Process -Filter "Name = 'notepad.exe'" |
select Name, Handle, ProcessId

[wmi]"Win32_Process.Handle=5464" |
select Name, Handle, ProcessId
```

wmilookup.ps1

```
Get-WmiObject Win32_Processor | select Name, Architecture | ft -a

$arch = DATA {
ConvertFrom-StringData -StringData @"
0 = x86
9 = x64
"@
}

Get-WmiObject Win32_Processor |
select Name,
```

```
@{Name="Architecture"; Expression={$sarch["$($_.Architecture)"]} } | ft -a
```

wmiVBScript.ps1

```
<#
set objWMIService = GetObject("winmgmts:" _
    & "{impersonationlevel=impersonate}!\\" _
    & ".\root\cimv2")

set colProcesses = objWMIService.ExecQuery _
    ("SELECT * FROM Win32_Process")

for each objProcess in colProcesses
    WScript.Echo " "
    WScript.Echo "Process Name : " + objProcess.Name
    WScript.Echo "Handle       : " + objProcess.Handle
    WScript.Echo "Total Handles: " + Cstr(objProcess.HandleCount)
    WScript.Echo "ThreadCount  : " + Cstr(objProcess.ThreadCount)
    WScript.Echo "Path         : " + objProcess.ExecutablePath
next
#>

Get-WmiObject Win32_Process | Select Name, Handle, HandleCount, ThreadCount,
ExecutablePath
```

wqlquery.ps1

```
$query = "SELECT * FROM Win32_Process"
Get-WmiObject -Query $query | select -first 5 | ft Name, ProcessID -a

#####

Get-WmiObject -Class Win32_Process | select -first 5 | ft Name, ProcessID -a
```