

Snel starten met PowerShell

- Ingebouwde help van powershell gebruiken
- Zie MSDN:
 - Win32 and COM Development
 - Administration and Management
 - Windows PowerShell
 - PowerShell Getting Started Guide
 - Using Windows PowerShell

Basisprincipe

- Powershell bevat functies en cmdlets
 - ▣ Uitgebreide help voor commando's, objecten,...
 - Help
 - ▣ Aliassen beschikbaar vb.
 - Set-Location heeft aliassen cd, chdir, sl
 - Get-Alias -definition Set-Location
 - ▣ Gebruik liever de volledige naam (beter leesbaar)
 - ▣ Completion: met TAB-toets

Opbouw cmdlets

- Syntax: Verb-noun
- Beperkt aantal werkwoorden (verbs):
 - ▣ get, select, move, ...
- Generisch: dezelfde woorden in andere context:
 - ▣ Move-Item : verplaats iets (file, registertak, environmentvariabele,...)
- Parameters toevoegen: start met -
 - ▣ Switchparameter: -recurse
 - ▣ met waarde: -name test

Consistent in gebruik

- Generisch: parameters met analoge functionaliteit hebben dezelfde naam voor alle cmdlets
- Gebruik * en ? in waarde – niet altijd ondersteund
- Hulp vragen over iets vb. cmdlet:
 - ▣ Get-Help cmdlet
 - ▣ Get-Help cmdlet -examples
 - ▣ Get-Help cmdlet -detailed
 - ▣ Get-Help cmdlet -full
 - ▣ Get-help cmdlet -parameter *
 - ▣ Get-help about*

Eerste voorbeelden

- Toon alle commando's:
 - ▣ Get-Command
- Toon hoe dit commando werkt:
 - ▣ Get-help Get-Command
- Toon de parameters voor dit commando
 - ▣ Get-help Get-Command -parameter *
- Toon de help voor een half-gekende parameter
 - ▣ Get-help Get-Command -parameter "*type*"
 - Merk op: Ondersteunt wild-cards

Twee belangrijke groepen cmdlets

- Get- cmdlets genereren meestal uitvoer
 - Get-Command -Commandtype alias
- Andere cmdlets bewerken uitvoer via |
 - Get-Command | Sort-Object
- Handige cmdlets, met aliasen:

Cmdlet	alias	Ook te schrijven als
Where-Object	where	?
ForEach-Object	foreach	%
Select-Object	select	
Sort-Object	sort	
Group-Object	group	

Objecten en lijsten

- Je kan uitvoer "opvangen" in een variabele.
 - \$lijst = Get-Command -Commandtype alias
- Toon de inhoud van de variabele
 - \$lijst
 - Write-Host \$lijst #met cmdlet
- Als er meer dan één object is wordt dit een lijst (een verzameling van objecten) met eigen properties en methodes (gebruik TAB)
 - \$lijst.count
 - (Get-Command).count #let op ()
- Variabele als invoer voor andere cmdlet via |
 - \$lijst | Sort-Object

Sort-Object

- Bekijk de parameters
 - Get-Help Sort-Object -parameter *
 - Get-Command | sort -descending
- Specificeer waarop geordend wordt:
 - Get-Command | Sort-Object -property CommandType, Name
 - \$lijst | Sort-Object CommandType, Name

Select-Object

- Bekijk de parameters:
 - Get-Help Select-Object -parameter *
- Enkel de eerste, laatste:
 - \$lijst | Select-Object -first 10
 - Get-Command | select -last 10
- Toon alle properties van één object uit de lijst
 - \$lijst | select -first 1 | select *
- Toon bepaalde properties van alle objecten
 - \$lijst | select -property Name
 - \$lijst | select Name
 - \$lijst | select -first 10 |select Name, Parameters

Where-object { \$_ ... }

- Beperking formuleren tussen { }
- Gebruik \$_ voor elk object uit de lijst
 - \$lijst | Where-Object{\$_ -like "*move*"}
- Gebruik de attributen van het object met .
 - \$lijst | where {\$_.CommandType -eq "Alias"}
- Je kan soms ook zoeken in multivalued attributen

Where-object { \$_ ... }

- Voorwaarden samenstellen met -and , -or

Comparison – operator	
-lt	Less than
-le	Less than or equal to
-gt	Greater than
-ge	Greater than or equal to
-eq	Equal to
-ne	Not equal to
-like	Like (uses wildcards for matching)
-notlike	Not like (uses wildcards for matching)

Group-Object

- Enkel zinvol met parameter
 - ▣ \$lijst | Group-Object -property CommandType
 - ▣ Get-Command | group CommandType
- Extra property count :
 - ▣ \$lijst | group Commandtype
 - | where {\$_.count -gt 100}

ForEach-Object

- iets doen voor elk object in de lijst
- Syntax


```
ForEach-Object {
    #doe iets met $_
}
```
- Voorbeeld:


```
Get-Service | foreach { $_.Name+" is "+$_Status}
```

Get-Member

- Bewerkt invoer (op variabele of na |)
- Toont beschikbare properties, methodes, ...
 - ▣ \$lijst | Get-Member
- Toont ook extra methodes of aliasen (o.a. converteren data)
- Beperk het overzicht
 - ▣ \$lijst | Get-Member -membertype property
- Vraag extra methodes/properties :
 - ▣ \$lijst | Get-Member -view all
- Geeft soms meerdere overzichten indien een lijst verschillende soorten objecten bevat
 - ▣ Vb Get-Command | Get-Member

Simple functions – positional parameters

get-help about_functions

```
function repeat {
    "Eerste parameter is " + $args[0]
    "Aantal parameters " + $args.count
    $args | foreach {
        "The input is $_"
    }
}
```

repeat een twee "drie en vier"

Simple functions – named parameters

get-help about_functions

Twee varianten :

```
function show
{
    param($name , $second)
    "name is "+$name
    "tweede is "+$second
}
```

```
show "Hallo Man" "dag"
show("Hallo Man","dag")
```

```
function show ($name , $second)
{
    "name is "+$name
    "tweede is "+$second
}
```

WMI en Powershell

▣ Met WMI-COM-objecten:

```
$Location = New-Object -comobject "WbemScripting.SWbemLocator"
$WmiService = $Location.ConnectServer(".", "root\cimv2")
$Klasse = $WmiService.get("Win32_LogicalDisk")
$Instantie = $WmiService.get("Win32_LogicalDisk.DeviceID='C:'")
```

▣ Met PS-WMI objecten - gebruik Get-WmiObject:

```
$PKlasse = Get-WmiObject -List "Win32_Directory" #Een klasse
$PInstantie = Get-WmiObject -class "Win32_LogicalDisk"
               | where {$_.DeviceID -eq 'C:'} #Een instantie
```

▣ Met eigen cmdlet (niet altijd beschikbaar)

- Get-Service
- Get-Process,
- ...

WMI-COM-objecten in Powershell

Propertes en methodes van

□ een WMI-Service:

- ▣ \$WmiService | get-Member
- ▣ \$WmiService | get-Member
| where{\$_Name -like "*Asso*"}
- ▣ | select Name,Definition | Format-List

SWbemServicesEx
+
SWbemServices

□ een klasse of instantie

- ▣ \$klasse | get-Member
- ▣ \$instantie | get-Member

SWbemObjectEx
+
SWbemObject

WMI-COM-objecten in Powershell

□ Klassequalifiers (klasse ≠ instantie)

- ▣ \$klasse.Qualifiers_ | select Name,Value | Format-List
- ▣ \$instantie.Qualifiers_ | select Name,Value | Format-List

□ Eén qualifier ophalen

- ▣ \$klasse.Qualifiers_.Item("abstract").Value
- ▣ if (\$klasse.Qualifiers_ | where{\$_Name -eq "abstract"})

Fout indien qualifier
niet bestaat

□ Alle klassequalifiers (enkel voor de klasse zinvol)

- ▣ \$klasse = \$WmiService.get("Win32_LogicalDisk", 131072)

Description

WMI-COM-objecten: methodes

□ Alle methodes en het aantal (klasse of instantie)

- ▣ \$klasse.Methods_ | select Name
- ▣ \$instantie.Methods_.Count

□ Eén methode ophalen:

- ▣ \$method = \$klasse.Methods_ | select -first 1
- ▣ \$method = \$klasse.Methods_.Item("Chkdsk")

□ Invoerparameters van een methode:

- ▣ \$method.InParameters.Properties_ | select Name,Value

WMI-COM-objecten: methode-qualifiers

□ Alle methode-qualifiers:

- ▣ \$method.Qualifiers_ | select Name,Value | Format-List

□ Ophalen van bepaalde qualifier:

- ▣ \$method.Qualifiers_.Item("Static").Value

Fout indien qualifier
niet bestaat

□ Enkel de statische methodes:

- ▣ \$klasse.Methods_ | where{\$_Qualifiers_.Item("Static")}

- ▣ \$klasse.Methods_ |
where{\$_Qualifiers_ | where{\$_Name -eq "Static"}}

WMI-COM-objecten: methode-qualifiers

□ Value-ValueMap verband is niet eenvoudig terug te construeren naar een hash...

- ▣ \$quals = \$method.Qualifiers_
- ▣ \$values=\$quals.Item("Values").Value
- ▣ \$valueMap=\$quals.Item("ValueMap").Value
- ▣ Zelf het verband vastleggen (zie oef)

Fout indien qualifier
niet bestaat

□ Resulteert in fout indien qualifier niet bestaat:

- ▣ if (\$quals | where{\$_Name -eq "Values"})

WMI-COM-objecten: attributen

□ Alle attributen (eigen+systeemattributen):

- ▣ \$klasse.Properties_ | select Name
- ▣ \$klasse.SystemProperties_ | select Name,Value
- ▣ \$instantie.Properties_.Count

□ Eén attribuut:

- ▣ \$klasse.Properties_.Item("VolumeName")

□ Meer informatie over attributen:

- ▣ \$klasse.Properties_ | Get-Member

□ Beperkte lijst attributen:

- ▣ \$klasse.Properties_ | where {\$_CIMTYPE -ne "8"}

WMI-COM-objecten: attribuutqualifiers

- Alle attribuutqualifiers van één attribuut:
 - ▣ \$klasse.Properties_.Item("VolumeName").Qualifiers_ | select Name, Value
- Een bepaalde attribuutqualifier:
 - ▣ \$klasse.Properties_.Item("Status").Qualifiers_.Item("Values").Value
 - ▣ \$klasse.Properties_.Item("Status").Qualifiers_.Item("ValueMap").Value

Resulteert in een fout indien de qualifier niet bestaat

WMI en Powershell

- Get-Help *wmi*
- Get-Help about_wmi*

Wmi-cmdlets:

Get-WmiObject	Gets instances of WMI classes or information about the available classes.
Invoke-WmiMethod	Calls WMI methods
Register-WmiEvent	Subscribes to a WMI event.
Remove-WmiObject	Deletes WMI classes and instances
Set-WmiInstance	Creates or modifies instances of WMI classes.

- ▣ WMI Type Accelerators :
[WMISEARCHER] [WMICLASS] [WMI]

Get-WmiObject

- Initialiseert objecten
 - ▣ Zoekt in default namespace root\cimV2
 - ▣ Parameter is verplicht (standaard -class)
 - ▣ Haalt alle instanties op van de opgegeven klasse
 - Get-WmiObject Win32_Service
- Aantal instanties:
 - ▣ (Get-WmiObject Win32_Service).Count
- Niet gebruiken met veel instanties!!

Get-WmiObject | select-Object

- Beperk de lijst (nadat ze volledig is opgehaald):
 - ▣ Get-WmiObject Win32_Service | select -first 1
- Beperk de attributen die getoond worden
 - ▣ Get-WmiObject Win32_Service | Select-Object Name, Status, State
- Geen onderscheid tussen soorten properties
 - ▣ Get-WmiObject Win32_Service | select Name, __RELPATH
- Ook array's worden direct getoond
 - ▣ Get-WmiObject Win32_Service | select -first 1 | select __DERIVATION

Get-WmiObject | where-Object

- Beschrijf een beperking met voorwaarden
- Je kan zoeken op alle attributen
 - ▣ Get-WmiObject Win32_Service | where {\$_.State -eq "Running"}
 - ▣ Get-WmiObject Win32_Service | where {\$_.__RelPath -like "*sql*"}
- Unieke instantie ophalen
 - ▣ \$Instantie = Get-WmiObject Win32_LogicalDisk | where {\$_.DeviceID -eq "C:"}

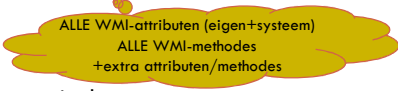
Get-WmiObject - parameters

- [-class] className :
 - ▣ alle instanties van die klasse (default parameter)
 - Get-WmiObject Win32_Service #-class is optioneel
 - ▣ Toont een aantal attributen voor elke instantie van die klasse
- -List [patroon] : enkel de klassen, geen instanties
 - ▣ Haalt altijd eerst de volledige lijst op - is traag
 - ▣ Optioneel patroon (wildcards mogelijk): beperkt de lijst
 - Get-WmiObject -List *Drive*
 - Get-WmiObject -List *
 - (Get-WmiObject -List).Count

Get-WmiObject - parameters

- ▣ **-query** WQLquery
 - Sneller dan where-Object
 - Niet combineren met **-class** of **-list** parameter
 - Get-WmiObject -query "select * from Win32_LogicalDisk"
 - Get-WmiObject -query "select * from Win32_LogicalDisk where DeviceID='C:'"
 - Gebruik | select om de gewenste attributen te tonen
- ▣ **-filter** WQLfilter_whereclause
 - Is steeds combinatie met **-class** parameter
 - Get-WmiObject Win32_LogicalDisk -filter "DeviceID='C:'"

Get-WmiObject

- ▣ Attributen, methodes bekijken:
 - ▣ Get-WmiObject Win32_Process | Get-Member
- 
- ▣ Data manipuleren:
 - ▣ \$PSinst = Get-WmiObject Win32_Process | select -last 1
 - ▣ \$ PSinst.ConvertToDateTime(\$ PSinst.CreationDate)

WMI-properties van PS-WMI-object

Bekijk dit met **Get-Member -memberType Property**

- ▣ Op instantie: alle WMI-properties (ook systeem-)
 - ▣ \$PSinstantie = Get-WmiObject -Class Win32_Process | select -first 1
 - \$PSinstantie | Get-Member -memberType Property
- ▣ Op klasse: enkel systeemproperties
 - ▣ \$PSklasse = Get-WmiObject -List Win32_Process
 - \$PSklasse | Get-Member -memberType Property


PS-properties van PS-WMI-object

Get-Member -memberType Property -view base

- ▣ Op klasse! (op instantie een kortere lijst)
 - \$PSklasse | Get-Member -memberType Property -view base
 - \$PSinstantie | Get-Member -memberType Property -view base

Name	Bevat
Methods	Alle WMI-methodes
Properties	Alle eigen WMI-properties
Qualifiers	Alle klasse-qualifiers
SystemProperties	Alle systeem-properties
...	

PS-WMI-object

- ▣ Klassequalifiers (op een klasse ≠ instantie)
 - ▣ \$PSklasse.Qualifiers | select Name,Value
- 
- ▣ Get-Help Get-WmiObject -parameter *
- \$PSklasse = Get-WmiObject -List Win32_Process
- \$PSklasse.amended.Qualifiers.Item("Description").Value

PS-WMI-object: methodes

- ▣ Alle methodes en het aantal (enkel op klasse)
 - ▣ \$PSklasse.Methods | select Name
 - ▣ \$PSklasse.Methods.Count
- ▣ Eén methode ophalen:
 - ▣ \$method = \$PSklasse.Methods | select -first 1
 - ▣ \$method = \$PSklasse.Methods.Item("Create")
- ▣ Invoerparameters van een methode geeft fout:
 - ▣ \$method.InParameters

PS-WMI-object: methode-qualifiers

- Alle methode-qualifiers:
 - ▣ `$method.Qualifiers | select Name,Value | Format-List`
- Ophalen van bepaalde qualifier:
 - ▣ `$method.Qualifiers.Item("ValueMap").Value`
 - ▣ `$PSKlasse.Methods | where{$_.Qualifiers.Item("Static")}`
- ▣ `$PSKlasse.Methods | where{$_.Qualifiers | where{$_.Name -eq "Static"}}`

Resulteert in een fout indien de qualifier niet bestaat

PS-WMI-object: attributen

- Alle attributen
 - ▣ `$PSKlasse.Properties | select Name`
 - ▣ `$PSKlasse.SystemProperties | select Name,Value`
 - ▣ `$PSInstantie.Properties.Count`
- Eén attribuut:
 - ▣ `$PSKlasse.Properties.Item("__RELPATH").Value`
 - ▣ `$PSKlasse.__RELPATH`
- Meer informatie over attributen (niet volledig):
 - ▣ `$PSInstantie.Properties | Get-Member`
- Beperkte lijst attributen:
 - ▣ `$PSKlasse.Properties | where {$_.CIMTYPE -ne "8"}`

PS-WMI-object: attribuutqualifiers

- Alle attribuutqualifiers van één attribuut:
 - ▣ `$PSKlasse.Properties.Item("Status").Qualifiers | select Name,Value`
- Een bepaalde attribuutqualifier:
 - ▣ `$PSKlasse.Properties.Item("Status").Qualifiers.Item("ValueMap").Value`

Resulteert in een fout indien de qualifier niet bestaat

PowerShell met/zonder WMI

- PowerShell heeft eigen cmdlets voor processen, services, mappen,...
- `Get-Process`, `Get-Service`, `Get-ChildItem`
 - Hebben eigen attributen en methodes
 - Heeft niets met WMI te maken, maar kan handig zijn
 - Zoek zelf uit of dit handiger is dan WMI-objecten

Tot slot

- Powershell is script-taal, beperkter dan perl
- Handige interface, waarmee je vlot kan experimenteren
- Repository doorzoeken is erg handig, maar de mogelijkheden blijven beperkt
- WMI-com-objecten gebruiken kan een eerste stap zijn voor een perl-script