Table des matières

I.Présentation

A.Préambule

Ce document est un support de cours dont l'objet est de fournir les clés de compréhension du PowerShell. Il ne peut pas faire l'objet de reproductions à des fins commerciales sans le consentement de son auteur.

B.Technologies de scripting

Tout système d'exploitation nécessite l'emploi de technologies complémentaires pour automatiser des tâches récurrentes. Unix et Linux disposent de différents shells. Avec Dos, puis Windows, Microsoft a développé différentes technologies de scripting. Initialement, il y a eu les commandes autour du DOS. Sous Windows NT, nous avions eu droit à Kix. Avec Windows, Bill Gates voulait faire de Visual Basic le langage universel. Nous avons eu droit à Vbscript utilisé dans Windows Scripting Host. Et puis, avec l'avènement de .Net, Microsoft a décidé de mettre en avant le PowerShell. Certains langages tels que Perl, Python présentent l'avantage de la portabilité.
Le PowerShell, d'un point de vue syntaxique, emprunte à différents langagestels que le Perl et aussi le Shell Unix.
La critique qu'on peut faire à Powershell est la lenteur de l'exécution due à l'utilisation du Framework .Net.

C.PowerShell 3

Windows PowerShell 3.0 nécessite Microsoft .NET Framework 4.0. La nouvelle version de PowerShell est disponible sur Windows 7 Service Pack 1, Windows Server 2008 R2 SP1 ou encore Windows Server 2008 Service Pack 2 par simple mise à jour. Elle est native sur Windows 8 et sur Windows Server 2012.
Pour déterminer la version de votre Powershell :

```
Get-Host | Select-Object Version
```

D.Les outils

- Windows PowerShell ISE, intégré à Windows 7
- Sapien's PrimalScript IDE
- PowerShell Scriptomatic
- Visual Studio
- Power GUI
- Pwoer Plus

II.Premiers pas

A.Les applets de commande ou cmdlets

Le langage PowerShell s'appuie sur un jeu de commandes qui peut être enrichi par l'installation de logiciels comme Microsoft Exchange 2007.

B.L'interpréteur

A partir de la ligne de commande, tapez *powershell* !

C.Protection

1.Le niveau de sécurité : Get-ExecutionPolicy

```
Get-ExecutionPolicy -List
```

2.Changer le niveau de sécurité : Set-ExecutionPolicy

Le paramètre *scope* permet de limiter le niveau de sécurité à l'utilisateur courant, à la machine, etc.

AllSigned	Seul les scripts "signés" fonctionnent
RemoteSigned	Les scripts locaux fonctionne, ceux d'internet doivent être "signés"
Restricted	Aucun script externe autorisé
Unrestricted	Aucune limite pour l'exécution des scripts

```
Set-ExecutionPolicy -Scope LocalMachine -ExecutionPolicy unrestricted
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy remotesigned
```

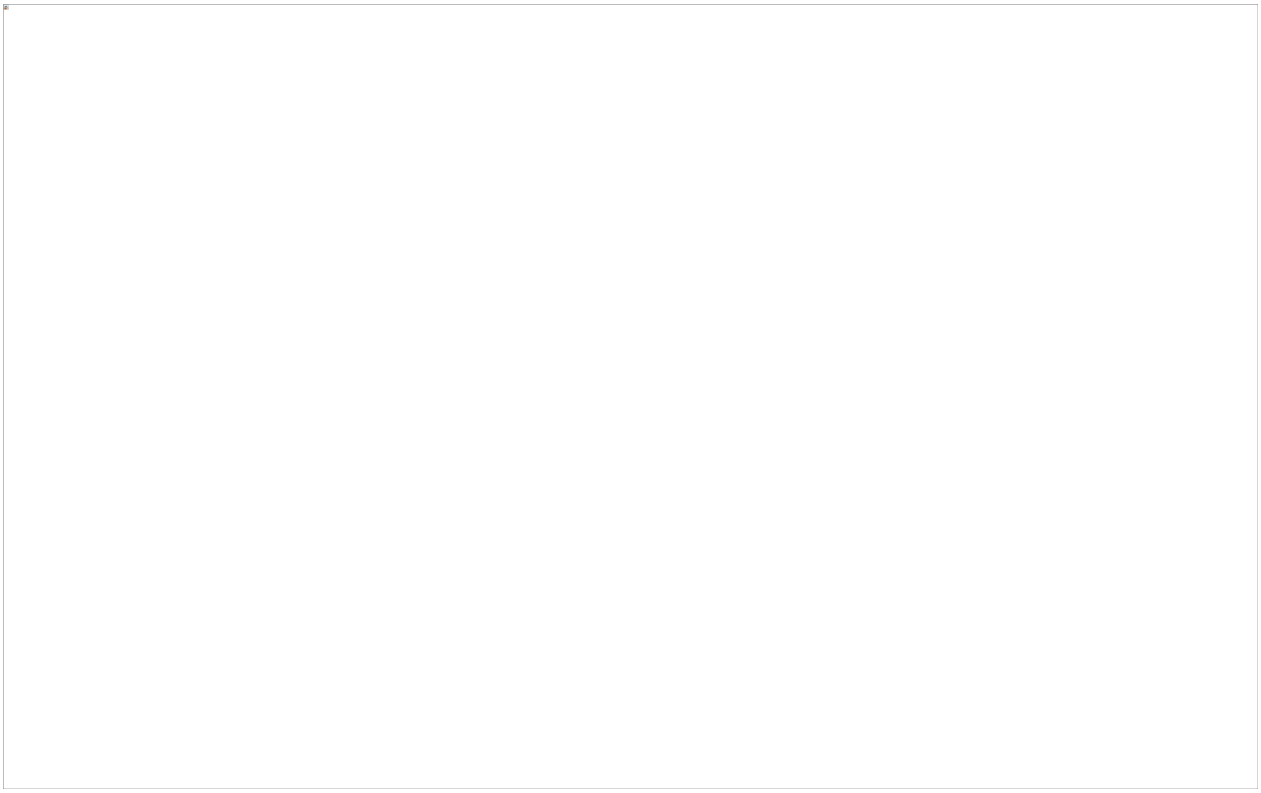
3.Signature

```
Get-AuthenticodeSignature "C:\windows\notepad.exe"
```

4.Voir aussi

```
GetHelp about_Execution_Policies
GetHelp about_Profiles
Get-ExecutionPolicy
Set-ExecutionPolicy
Set-AuthenticodeSignature
```


5.Stratégies



6.Autorité de certification

La commande makecert.exe est installée avec Office ou Visual Studio.
makecert.exe -n "CN=Dsfc" -a sha1 -eku 1.0 -r -sv private.pvk certificat.cer -ss Root -sr localMachine

7.Associer un certificate à un script

```
$cert=@(Get-ChildItem cert:\Currentuser\My)[0]
Set-AuthenticodeSignature d:\test.ps1 $cert
```

D.Aide

1.Informations de plate-forme : *Get-Host*

Get-Host fournit, notamment, la version du PowerShell.

2.La liste des commandes : *Get-Command*

3.L'aide : Get-Help

```
Get-Help about
Get-Help Set-Service –examples
get-help Set-Service –detailed
get-help Set-Service -full
Get-Help Set-Service –online
Get-Help *Service*
Get-Help *s* -Category Alias
Get-Command -Verb Get
Get-Command -Module NetTcpIp
Get-Help * -Parameter ComputerName
```


4.Actualiser l'aide 3.0

Update-Help

5.Méthodes et propriétés associées à une cmdlet

```
Get-Date|Get-Member
Get-Date | Get-Member -membertype methods
Get-Date | Get-Member -membertype properties
Get-Process | Get-Member -membertype aliasproperty
(Get-Process).ProcessName
(Get-Host).CurrentCulture | format-list -property *
(Get-Host).CurrentCulture.TextInfo.ANSICodePage
Get-Process | Sort-Object -Property CPU
Get-Process | Sort-Object -Property CPU -Descending
Get-Process | Sort CPU
```

6.Afficher les propriétés d'un cmdlet

```
Get-Process |Select-Object ProcessName,PrivateMemorySize
```

7.Mode GUI

```
Show-Command
Show-Command -Name Get-Process
```

8.Afficher les méthodes et propriétés d'un objet

L'utilisation du connecteur MySQL .Net suppose que vous l'ayez téléchargé et installé au préalable.

```
[void][system.reflection.Assembly]::LoadFrom("C:\Program Files\MySQL\MySQL
Connector Net 6.3.6\Assemblies\v2.0\MySql.Data.dll")
New-Object MySql.Data.MySqlClient.MySqlConnection | Get-Member
```

9.Les fournisseurs PowerShell : Get-PSProvider

Get-PSProvider
Get-ChildItem Env:
Set-Location Env:
New-Item –Name Test –Value 'Mon test à moi'
Get-Content Env:Test
Remove-Item Env:Test

10.Historique

Start-Transcript
Stop-Transcript

E.Exécution des scripts

1.Exécution d'un script

```
powershell d:\scripts\monscript.ps1
```

2.Appel d'un autre script

```
Invoke-Expression d:\scripts\monscript.ps1
& d:\scripts\monscript.ps1
d:\scripts\monscript.ps1
Invoke-Expression "d:\ scripts\monscript.ps1"
```


3.Récupération du contenu de l'exécution d'une commande système

```
clear
$res=&hostname
$res.Trim
#$res=&{. 'c:\windows\system32\ipconfig.exe'}
$res=&'c:\windows\system32\ipconfig.exe'
clear
If($res|where {$_ -match 'IPv4[ \.]+:[ ]+(\d+\.\d+\.\d+\.\d+)'})
{
    $Matches[1]
}
```

4.Mac Address

```
clear
$cmd=&c:\windows\system32\ipconfig.exe /all
#$cmd[10]
$cmd|Foreach{
    if($_ -match '([0-9a-f\-]{17})')
    {
        $matches[1]
        break
    }
}
```

5.Variable d'environnement

```
Foreach($item in (Get-ChildItem env:\))
{
    "$($item.Key) : $($item.value)"
}
$env:COMPUTERNAME
```

6.Appel d'un programme

```
Invoke-Item c:\windows\system32\calc.exe
```

7.Mesurer le temps d'exécution : Measure-Command

```
Clear
Write-Output "Ceci est un test"
$temps=Measure-Command { sleep -Seconds 1}
Write-Output "Mesure n°1: $temps"
$temps=Measure-Command {Write-Output "La commande est exécuté. Le message n'est
pas affiché." }
Write-Output "Mesure n°2: $temps"
$temps=Measure-Command {Write-host "La commande est exécuté. Et, cette fois, vous
pouvez le voir." }
Write-Output "Mesure n°3: $temps"
Measure-Command {d:\scripts\monscript.ps1}
```

8.Tempo

```
Start-Sleep -s 10
Start-Sleep -m 10000
```

9.Trigger

```
$DailyTrigger = New-JobTrigger -At 17:25 -Daily
Register-ScheduledJob -Name RestartFaultyService -ScriptBlock {Restart-Service
FaultyService }-Trigger $DailyTrigger
Get-ScheduledJob
```



```
Get-ScheduledJob -Name RestartFaultyService
Disable-ScheduledJob -Name RestartFaultyService
Enable-ScheduledJob -Name RestartFaultyService
Unregister-ScheduledJob -Name RestartFaultyService
```

10.Envoi de mail

a)Méthode Send-MailMessage

```
$motdepasse = ConvertTo-SecureString "denis" -AsPlainText -Force
$authentification = New-Object System.Management.Automation.PSCredential
("denis@dutout.net", $motdepasse)
#Get-Credential -UserName 'denis@dutout.net' -Message Denis
Send-MailMessage -To 'denis@dutout.net' -Subject 'test PS' -From 'denis@-
dutout.net' -Body 'test PS' -SmtpServer 'smtp.dutout.net' -Credential $authentifi-
cation
```

Méthode .Net

```
$CredUser = "dszalkowski"
$CredPassword = "areuhhh"
$EmailFrom = "dszalkowski@gmail.com"
$EmailTo = "dszalkowski@gmail.com"
$Subject = "Test PS2"
$Body = "Test PS2"
$SMTPServer = "smtp.gmail.com"
$SMTPClient = New-Object Net.Mail.SmtpClient($SmtpServer, 587)
$SMTPClient.EnableSsl = $true
$SMTPClient.Credentials = New-Object System.Net.NetworkCredential($CredUser,
$CredPassword);
$SMTPClient.Send($EmailFrom, $EmailTo, $Subject, $Body)
```

F. Historique

1.Visualiser l'historique

```
Get-History
Get-History 32 -count 32
$MaximumHistoryCount = 150
```

2.Récupérer l'historique

```
Get-History | Export-Clixml "d:\scripts\my_history.xml"
Import-Clixml "d:\scripts\my_history.xml" | Add-History
```

3.Exécuter une commande de l'historique

```
Invoke-History 3
```

4.Voir aussi

```
about_history
Invoke-History
Add-History
Clear-History
```

G.Informations de langue

```
Get-Culture
Get-UICulture
```


H.Passage d'arguments

1.Par tableau

```
$res=0
foreach($argument in $args)
{
	Write-Host $argument
}
```

2.Par la méthode Param

```
./monscript.ps1  -path "c:\windows" -value 1
Param ([string]$path, [int]$value)
Write-host   "le chemin est : $path et la valeur est : $value"
```

I.Commentaires

Commenter une ligne : #
Commenter un bloc : <# … #>

J.Instruction sur plusieurs lignes

III.Cmdlets système

A.Le journal d'événements

```
Get-EventLog -list
Get-EventLog -list | Where-Object {$_.logdisplayname -eq "System"}
Get-EventLog system -newest 3
Get-EventLog -LogName application | where entrytype -eq 'error'
```

B.Les services

1.La liste des services

```
Get-Service
Get-Service | Where-Object {$_.status -eq "stopped"}
Get-Service | Where-Object {$_.status -eq "running"} |Select-Object Name,
DisplayName
Get-Service | Sort-Object status,displayname
Get-Service | Sort-Object status | Group-Object -Property status
```

2.Démarrer, arrêter un service

```
Stop-Service MySQL
Start-Service MySQL
Restart-Service MySQL
Restart-Service -displayname "MySQL"
```

3.Mettre en suspens, reprendre un service

Le service en état suspendu ne permet plus des connexions supplémentaires.

```
Suspend-Service MySQL
Resume-Service tapisrv
```

4.Modifier les propriétés des services

```
set-service -name lanmanworkstation -DisplayName "LanMan Workstation"
get-wmiobject win32_service -filter "name = 'SysmonLog'"
set-service sysmonlog -startuptype automatic
Startuptype : manual, stopped
Set-Service clipsrv -startuptype "manual"
Set-Service "ati hotkey poller" -description "This is ATI HotKey Poller service."
```

C.Les process

1.Liste des process

```
Get-Process
Get-Process winword
Get-Process winword,explorer
Get-Process w*
Get-Process | Select-Object name,fileversion,productversion,company
Get-Process | Where-Object WorkingSet -gt 100MB | Select-Object Name
Get-Process | sort name | group name -NoElement | sort count –Descending
Get-Process | Where { $_.starttime.minute -lt 30} | select name, starttime
```


2.Arrêter un process

```
Stop-Process 3512
Stop-Process -processname notepad -Verbose
Stop-Process -processname note*
```

3.Verbosité/Erreur

```
Stop-Process -processname notepad -Verbose
Get-Process -Name notepad -ErrorAction SilentlyContinue
```

D.Informations

```
Get-Host
Get-Hotfix
Get-HotFix|Where InstalledOn -lt 2/9/2013
```

E.CIM

Get-CIMClass –Class *network*

```
(Get-CimClass -Class Win32_NetworkAdapterConfiguration).CimClassMethods
(Get-CimClass -Class Win32_NetworkAdapterConfiguration).CimClassProperties
Get-CimClass -PropertyName speed
Get-CimClass -MethodName reboot
Get-CimClass -Class Win32_BIOS
Get-CimInstance -ClassName Win32_BIOS
(Get-CimInstance -ClassName Win32_BIOS).SerialNumber
```

F.WMI

```
Get-WmiObject -List
Get-WmiObject win32_bios
Get-WmiObject win32_bios -computername atl-fs-01
Get-WmiObject win32_bios | Select-Object *
Get-WmiObject win32_bios | Select-Object -excludeproperty "_*"
$data = Get-WmiObject Win32_OperatingSystem
$share = Get-WmiObject Win32_Share
$cpu = (Get-WmiObject win32_processor | select-object
loadpercentage).loadpercentage
$availMem =( Get-WmiObject win32_perfFormattedData_perfos_memory | select-object
availableMbytes).availableMBytes / 1024
```


IV.Eléments du langage

A.Les variables et les constantes

1.Les variables

```
$Mem= WmiObject Win32_ComputerSystem
$Mbyte =1048576 # Another variable
"Memory Mbyte " + [int]($Mem.TotalPhysicalMemory/$Mbyte)
[int]$a =7
$a +3
$a
$DriveA, $DriveB, $DriveC, $DriveD = 250, 175, 330, 200
$i=0
[string]$Type = "Win32"
$WMI = Get-wmiobject -list | Where-Object {$_.name -match $Type}
Foreach ($CIM in $WMI) {$i++}
Write-Host 'There are '$i' types of '$Type
```

2.Les types

'Texte' –is [string]
$a = 55.86768
$b = $a.GetType().name

3.Les chaînes

Les chaînes de caractère peuvent être encadrées de guillemets ou d'apostrophes.
Les guillemets peuvent interpréter des variables

```
$a='test'
$b="$a"
write-Output $b
#Here-String
$texte=@'
hgfhgh
gjgjjgj
'@
```

4.Caractères spéciaux

`0	Null
`a	Beep
`b	Backspace
`n	Saut de ligne
`r	Retour chariot
`t	Horizontal tab
`'	Single quote
`"	Double quote
`f	Saut de page
`v	Tabulation verticale

5.Substitution de variables

```
$fichier=Get-ChildItem c:\windows\system32\drivers\etc\services
$l=$fichier.Length
$n=$fichier.FullName
```



```
clear
"Taille du fichier $n : $l octets"
"Taille du fichier {1} : {0} octets" -f $l,$n
```

6.Les variables prédéfinies

$$	Dernière commande
$?	True si la commande a réussie / False si échouée
$Args	Tableau des paramètres passés à partir de la ligne de commande
$ConsoleFileName	Chemin du dernier fichier utilisé dans la session
$Error	Liste des erreurs de la session
$Event	Evénement traité par Register-ObjectEvent
$EventArgs	Arguments relatifs à Event
$Foreach	Enumerateur d'une boucle ForEach
$Home	Répertoire de base de l'utilisateur
$Host	Informations sur l'hôte
$LastExitCode	Code de sortie de la dernière commande du système execute
$PID	Process du script PowerShell
$Profile	Chemin du profil PowerShell
$PSHome	Répertoire d'installation du PowerShell
$PSItem ou $_	Objet courant
$PSScriptRoot	Répertoire du script
$PSVersionTable	Information sur PowerShell
$PWD	Répertoire courant
$ShellID	Identificateur du Shell
$MyInvocation	`$MyInvocation.MyCommand.Name`

Les constantes

```
Set-Variable Thermometer 32 -option constant.
Set-Variable AllOverPlace 99 -scope global
$global:runners = 8
$alert = Get-Service NetLogon
$alert.status
```

B.Les tableaux

1.Principes de base

L'indice d'un tableau commence à 0.
$tab=1,2,3,4
$tab=0..99
$Jours="Lu","Ma","Me","Je","Ve","Sa","Di"
[int[]]$tab=1,2,3,4
$tab=[string]'Texte',[int]8,[double]3.47,[char]'z'
$tab[0] Lit le 1er élément du tableau
$tab[$tab.length-1] Dernier élément du tableau
$tab.length Nombre d'éléments du tableau
$tab[0..2] Affiche les éléments de l'indice 0 à 2
$tab[-1] Dernier élément
$tab1+$tab2 Concaténation de tableau
$tab+=4 Ajout d'un élément au tableau
Pas de suppression de tableau
$tab=1,2,3,4
$tab=$tab[0..1+3]
$tab=$tab|Where-Object {$_ -ne 3}

Exemple

```
clear
[string[]]$Jours='Lu','Ma','Me','Je','Ve','Sa','Di'
$Jours[0]
$Jours[-1]
$jours.Length
$jours+='Dredi'
$Jours[-1]
#$Jours=$Jours|Sort
#$Jours=$Jours[0..4+7]
$Jours=$Jours|Where {$_ -match 'e'}
clear
$Jours
```

2.Effacer un élément avec méthode .Net

```
Clear
$a = New-Object System.Collections.ArrayList
$a.Add("red")
$a.Add("yellow")
$a.Add("orange")
$a.Add("green")
$a.Add("blue")
$a.Add("purple")
$a.Remove("yellow")
$a
$a=$null
```

3.Tableaux associatifs

```
$recettes=[ordered]@{Lu=100;Ma=800;Me=350;Je=560;Ve=340}
$recettes|Format-List
$recettes['Ve']
$recettes+=@{Sa=1230}
$recettes.keys
$recettes.values
$recettes.keys|Foreach {$recettes.$_}
```

4.Autres méthodes

```
Set-Variable server -option None -force
Set-Variable server -option Constant -value '10.10.10.10'
Remove-Variable server -force
```

5.Portée

$global:variable	Par défaut
$local:variable	Locale à la function, au script, au bloc d'instructions
$script:variable	Script
$using:variable	Exécution à distance

Nombre aléatoire

```
(New-Object system.random).next()
Get-Random
Get-Random -Maximum 21 -Minimum 1
Get-Random -InputObject (1..10) -Count 5
```

C.Opérateurs

1.Concaténation

+

2.Comparaison

-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 pattern matching
-match Expression régulière
1 –lt 2

3.Expressions régulières

'PowerShell' –match 'l$'
'PowerShell' –notmatch 'l$'
$Matches, $Matches[i]
'Date: 02/09/2013' –match '^Date:\s(?<date>(?<jour>\d{2})/>(?<mois>\d{2})/>(?<annee>\d{4}))$'
$Matches.annee

```
clear
$Str="Henri est au boulot avec Denis"
$Regex="(Henri)( est au boulot avec )(Denis)"
$new=$Str –replace $Regex, '$3$2$1'
$new
$Str=$null;$Regex=$null
```

4.Logiques

-and Et
-or Ou
-xor Ou exclusif

5.Plages

1..99

6.Appartenance

'D' -in 'DSFC','Szalkowski'
'D' -notin 'DSFC','Szalkowski'
Contains, c'est l'inverse : 'DSFC','Szalkowski' contains 'D'

7.Opérateurs binaires

-band
-bor
-bnot
-bxor

8.Affectation

$i=0
$i++
$i=$i+8 ou $i+=8

9.Cast

```
clear
$b=Read-Host 'Saisissez votre élément'
if($b -match '^\d+$')
{
```



```
    $b=[int]$b
    $b*100
}
else
{
    'Ceci n''est pas une valeur'
}
$b.GetType().Name
```

10.Forcer la définition de variables

Set-PSDebug -Strict

D.Structures de contrôle

1.Do

```
$a = 1
do {$a; $a++}
while ($a -lt 10)
$a = 1
do {$a; $a++} until ($a -eq 10)
```

2.While

```
$a = 1
while ($a -lt 10) {$a; $a++}
```

3.For

```
for ($a = 1; $a -le 10; $a++) {$a}
```

4.Break

```
$a = 1,2,3,4,5,6,7,8,9
foreach ($i in $a)
{
    if ($i -eq 3)
    {
        break
    }
    else
    {
        $i
    }
}
```

5.If

```
$a = "white"
if ($a -eq "red")
    {"The color is red."}
elseif ($a -eq "white")
    {"The color is white."}
else
    {"The color is blue."}
```

6.Foreach

```
Foreach ($item in Get-Process)
{
```



```
    "$($item.CPU*1000)"
}
Get-Process|Foreach{
    "$($_.CPU*1000)"
}
Get-Process|Foreach{$_.CPU*1000}
Get-Process|Foreach CPU
foreach ($i in get-childitem c:\windows)
{$i.extension}
"un vélo.","un ballon","une chouette." | ForEach-Object Insert
-ArgumentList 0,"C'est "
```

7.Switch

```
$a = 5
Switch ($a)
    {
        1 {"The color is red."}
        2 {"The color is blue."}
        3 {"The color is green."}
        4 {"The color is yellow."}
        5 {"The color is orange."}
        6 {"The color is purple."}
        7 {"The color is pink."}
        8 {"The color is brown."}
        default {"The color could not be determined."}
    }
Switch -regex (chaine)
{
'^test'{'Ca commence par test';break}
'test$' {'Ca finit par test';break}
}
```

8.Exemple conditionnelle

```
Clear
$chaine=Read-Host 'Texte'
Switch -regex ($chaine)
{
'^test'{'Ca commence par test';break}
'test$' {'Ca finit par test';break}
Default {'Ni l''un, ni l''autre'}
}
If($chaine -Match '^test')
{
    'Ca commence par test'
}
ElseIf($chaine -Match 'test$')
{
    'Ca finit par test'
}
Else
{
    'Ni l''un, ni l''autre'
}
```

E.Gestion d'erreurs

1.Préférence

$ErrorActionPreference='SilentlyContinue'
Valeurs possibles : SilentlyContinue, Continue, Stop, Inquire, Ignore (3.0 : non stockée dans $Error)

2.Cas par cas

```
Get-ChildItem c:\test.bat -ErrorAction SilentlyContinue -ErrorVariable err
$err
```

Trap

```
clear
$ErrorActionPreference='SilentlyContinue'
trap { 'Erreur';exit}
100/0
Get-Process
```

Try...Catch

```
clear
Try
{
    100/0
}
Catch
{
    "Errare humanum est, sed...`n$($Error[0])"
}
Finally
{
    'J''ai fait mon boulot'
}
```

Débogage

$VerbosePreference
Write-Verbose
Write-Debug
Set-PSDebug –Step
Set-PsBreakPoint –Command Get-Process : point de débogage à chaque exécution de la commande Get-Process
Commandes Débogeur : S (Suivant et retour),V,O,L,G (Stop),K (Pile)

F.Pipelining

1.Comptage

```
Get-Service | Group-Object status
Get-ChildItem c:\windows | Group-Object extension
Get-ChildItem c:\windows | Group-Object extension | Sort-Object count
```

2.Stats

```
Get-Process | Measure-Object CPU -ave -max -min -sum
```

3.Sélection

```
Get-Process|Select-Object ProcessName -first 5
```

4.Tri

```
Get-Process|Select-Object ProcessName, Id |Sort-Object Id
```


5.Différence

a)Process

```
Clear
$A = Get-Process
Stop-Service MySQL
$B = Get-Process
Start-Service MySQL
Compare $A $B
```

b)Fichiers

```
$A = Get-Content d:\scripts\x.txt
$B = Get-Content d:\scripts\y.txt
Compare-Object A$ B$
```

6.Affichage

a)Liste

```
Get-Service | Format-List -Property
Get-Service | Format-List *
```

b)Tableau

```
Get-Service|Format-Table
Get-Service | Where Status -eq 'Running'| Format-Table -Property Name,DisplayName
Get-Service | Where Status -eq 'Running'| Format-Table -Property Name,DisplayName
-GroupBy Name
Get-Service | Where Status -eq 'Running'| Format-Table -Property Name,DisplayName
-AutoSize
```

c)Colonne

```
Get-Service|Format-Wide –Property Name –autosize
Get-Service|Format-Wide –Property Name –column 4 –autosize
```

d)Write-Output

C'est la commande implicite

```
Get-Eventlog PowerShell | Out-Host -paging
Get-Eventlog PowerShell | Out-Host -p
Get-Eventlog PowerShell | more
```

e)Write-Host

Il renvoie vers la console et ne peut pas renvoyer vers un fichier

f)Exemples

```
Get-Service|Where Status -eq 'Running'|Select Name,DisplayName|Format-Table -Au-
toSize -HideTableHeaders
Get-Process|Where-Object { $_.Name -match '^S'}|Select Name,Handle|Format-List
-GroupBy Name
Get-Process|Out-GridView -Title 'Mon bô tableau, roi des ...'
```


7.Filtre

a)Avec Where-Object

```
Get-Service|Where-Object {$_.Status -eq 'Running'}|Select-Object Name,
DisplayName|Format-Table –autosize
Get-ChildItem c:\windows|Where-Object {$_.Name -like '*.exe'}|Select-Object Name
```

b)Avec filter

```
Filter Get-BigProcess
{
    Begin
    {
        $conso=0
    }
    Process
    {
        If($_.CPU -gt 1)
        {
            $_
        }
        $conso+=$_.VM
    }
    End
    {
        "`nConso cumulée des process de plus de 100MB : $($conso/(1024*1024)) Mo"
    }
}
Get-Process|Get-BigProcess
```

8.Valeurs unique

```
Get-Content d:\scripts\test.txt | Sort-Object | Get-Unique
Get-Process|Sort-Object ProcessName|Get-Unique|Select-Object ProcessName
Get-Process|Select Name|Sort|Get-Unique -AsString
Get-Process|Select Name|Sort Name -Unique
```

9.Propriétés

```
Get-ItemProperty  "hklm:\SYSTEM\CurrentControlSet\services\MySQL"
```

10.Impressions

```
Get-Process | Output-Printer
Get-Process | Output-Printer "HP LaserJet 6P"
```

11.Boucle

```
Get-Process | Where Handle -gt 0
Get-Process | Where-Object Handle -gt 0
Get-Process |ForEach-Object {Write-Host $_.ProcessName -foregroundcolor cyan}
#$rows = get-wmiobject -class Win32_QuickFixEngineering
#foreach ($objItem in $rows)
#{
#      write-host "HotFix ID: " $objItem.HotFixID
#}
#get-wmiobject -class Win32_QuickFixEngineering|Select-Object HotFixID
get-wmiobject -class Win32_QuickFixEngineering|ForEach-Object {Write-Host
$_.HotFixID}
```


12.Tri

```
Get-ChildItem c:\windows\*.* | Sort-Object length -descending | Select-Object
-first 3
Get-EventLog system -newest 5 | Sort-Object eventid
```

13.Message

```
Write-Warning "The folder D:\scripts2 does not exist."
Write-Host "This is red text on a yellow background" -foregroundcolor red
-backgroundcolor yellow
```

a)Couleurs

Black
DarkBlue
DarkGreen
DarkCyan
DarkRed
DarkMagenta
DarkYellow
Gray
DarkGray
Blue
Green
Cyan
Red
Magenta
Yellow
White

14.Interaction

```
$Name = Read-Host "Please enter your name"
Write-Host $Name
```

G.Fonctions

1.Sans retour

```
Function Set-Popup
{
    param([string]$title,[string]$message)
    $oWsh=New-Object -ComObject Wscript.shell
    $oWsh.Popup($message,0,$title)
}
Set-Popup -title 'Ma boîte à moi' -message 'Mon texte à moi'
```

2.Avec retour

```
Function Conso-Memoire
{
    Param([string]$process)
    Get-Process|Foreach{
        if($process -eq $_.ProcessName)
        {
            [math]::round($_.VM/1048576)
            break
        }
    }
    0
```



```
}
Conso-Memoire -process 'firefox'
. 'C:\powershell\biblio.ps1'
Get-DriveFreeSpace -Letter 'c:'
```

H.Gestion des modules

1.Emplacement des modules

Ils sont déterminés par la variable d'environnement `$env:PSModulePath`.
%windir%\System32\WindowsPowerShell\v1.0\Modules
%UserProfile%\Documents\WindowsPowerShell\Modules

2.Télécharger des modules complémentaires

http://gallery.technet.microsoft.com/scriptcenter/site/search?f[0].Type=ProgrammingLanguage&f[0].Value=PowerShell&f[0].Text=PowerShell&sortBy=Downloads

3.Les modules liés à l'administration

```
Get-Module -ListAvailable  Liste tous les modules
```

4.Commandes d'un module

```
Get-command –module DnsServer
```

5.Charger automatiquement les modules

$PSModuleAutoloadingPreference='All' (None,ModuleQualified)

6.Décharger un module

Remove-Module Dnsserver

7.Créer un module

Créez un répertoire et un fichier psm1 du même nom dans l'un des répertoires défini par `$env:PSModulePath`

8.Exemple : devices.psm1

a)Définition des fonctions du module

```
<#
    .Synopsis
    Indique le taux d'espace libre.

    .Description
    La fonction Get-DriveFreeSpace indique le taux d'espace libre
    calculé à partir de l'appel à WMI.
    .Parameter Letter
    Entrez la lettre de lecteur telle que C:.

    .Example
    Get-DriveFreeSpace 'C:'
    .Example
    Get-DriveFreeSpace -Letter 'C:'
    .Link
    Get-DriveFreeSpace
#>
Function Get-DriveFreeSpace
{
    Param([string]$Letter)
```



```
    #$res=$null
    $Drives=Get-WmiObject Win32_LogicalDisk| Where Size -ne $Null
    Foreach($Drive in $Drives)
    {
        If($Drive.DeviceID -eq $Letter)
        {
            $res=[Math]::Round($Drive.FreeSpace/$Drive.Size*100,2)
            Return $res
            #Break
        }
    }
    #$res
}
```

b) Utilisation du module

```
Import-Module Devices
Devices\Get-DriveFreeSpace -Letter 'D:'
$env:PSModulePath
```


V.Gestion des heures et des dates

A.Obtenir la date et l'heure : Get-Date

```
Get-Date
Get-Date -displayhint date
Get-Date -displayhint time
$Date=Get-Date –Year 2013 –Month 9 –Day 1
$A = Get-Date 5/1/2006
$A = Get-Date "5/1/2006 7:00 AM"
(Get-Date).AddMinutes(137)
$date = Get-Date -Format 'dd-MM-yyyy'
Get-Date -format 'yyyyMMddHHmmssfff'
Get-Date –Format d
Formats : d, D,f,F,g,G,m,M,r,R,s,t,T,u,U,y,Y
```

B.Méthodes associées à la cmdlet Get-Date

```
AddSeconds
AddMinutes
AddHours
AddDays
AddMonths
AddYears
```

C.Changer la date et l'heure : Set-Date

```
Set-Date -date "6/1/2006 8:30 AM"
Set-Date (Get-Date).AddDays(2)
Set-Date (Get-Date).AddHours(-1)
Set-Date -adjust 1:37:0
(Get-Date).addYears(1).dayOfWeek
([DateTime]'01/21/1964').DayOfWeek
```

D.Calculs sur date

```
New-TimeSpan $(Get-Date) $(Get-Date -month 12 -day 31 -year 2006)
$(Get-Date)
New-TimeSpan $(Get-Date) $(Get-Date -month 12 -day 31 -year 2006)
New-TimeSpan $(Get-Date) $(Get-Date -month 12 -day 31 -year 2006 -hour 23 -minute
30)
New-TimeSpan  $(Get-Date 1/1/2011) $(Get-Date 31/12/2011)
```

E.Création de fichier

New-Item –Type file –Name "Rapport_$((Get-Date –Format 'yyyyMMdd')).txt"

VI.Gestion des fichiers

PowerShell propose les mêmes commandes pour manipuler le système de fichiers et la base de registre.

A.Système

1.Copie de fichiers : Copy-Item

```
Copy-Item d:\scripts\test.txt c:\test
Copy-Item d:\scripts\* c:\test
Copy-Item d:\scripts\*.txt c:\test
Copy-Item d:\scripts c:\test -recurse
```

2.Création de fichiers : New-Item

```
New-Item d:\scripts\Windows PowerShell -type directory
New-Item d:\scripts\new_file.txt -type file
New-Item d:\scripts\new_file.txt -type file -force
```

3.Déplacer les fichiers

```
Move-Item d:\scripts\test.zip c:\test
Move-Item d:\scripts\*.zip c:\test
Move-Item d:\scripts\test.zip c:\test -force
Move-Item d:\scripts\950.log c:\test\mylog.log
```

4.Renommer les fichiers

```
Rename-Item d:\scripts\test.txt new_name.txt
```

Suppression de fichiers : Remove-Item

```
Remove-Item d:\scripts\test.txt
Remove-Item d:\scripts\*
Remove-Item d:\scripts\* -recurse
Remove-Item c:\*.tmp –recurse
Remove-Item d:\scripts\* -exclude *.wav
Remove-Item d:\scripts\* -include .wav,.mp3
Remove-Item d:\scripts\* -include *.txt -exclude *test*
```

B.Informations sur les fichiers, répertoires et clés de registres

```
$(Get-Item c:\).lastaccesstime
$(Get-Item hklm:\SYSTEM\CurrentControlSet\services).subkeycount
```

C.Tester l'existence d'un chemin

```
Test-Path d:\scripts\test.txt
Test-Path d:\scripts\*.wma
Test-Path HKCU:\Software\Microsoft\Windows\CurrentVersion
```

D.Lire un répertoire

1.Commandes

```
Get-ChildItem -recurse
Get-ChildItem HKLM:\SYSTEM\CurrentControlSet\services
Get-ChildItem d:\scripts\*.* -include *.txt,*.log
```



```
Get-ChildItem d:\scripts\*.* | Sort-Object length
Get-ChildItem d:\scripts\*.* | Sort-Object length -descending
Get-ChildItem | Where-Object { -not $_.PSIsContainer } : liste les fichiers
uniquement
Get-ChildItem -File : idem à la précédente
Get-ChildItem -Force | Where-Object { -not $_.PSIsContainer -and $_.Attributes
-band [IO.FileAttributes]::Archive }
Get-ChildItem -File -Hidden : idem à la précédente
Get-ChildItem -Attribute !Directory+Hidden,!Directory
```

2.Attributs (IO.FileAttributes)

- ReadOnly
- Hidden
- System
- Directory
- Archive
- Device
- Normal
- Temporary
- SparseFile
- ReparsePoint
- Compressed
- Offline
- NotContentIndexed
- Encrypted

E.La sécurité

```
Get-Acl d:\scripts | Format-List
Get-Acl HKCU:\Software\Microsoft\Windows
Get-Acl d:\scripts\*.log | Format-List
$acls=Get-Acl -Path 'c:\test\fictest.txt'
ForEach($fic in Get-ChildItem 'd:\powershell')
{
    $path=$fic.FullName
    Set-Acl -Path $path -AclObject $acls
}
```

F.Ajout à un fichier

```
Add-Content d:\scripts\test.txt "The End"
Add-Content d:\scripts\test.txt "`nThe End"
```

G.Recherche dans un fichier

```
Select-String -Path 'c:\windows\ntbtlog.txt' -Pattern 'Did not load driver'
Select-String -Path 'c:\windows\ntbtlog.txt' -Pattern 'Did not load driver' -List
Select-String -Path 'c:\windows\ntbtlog.txt' -Pattern 'Did not load driver'
-quiet
Get-Content d:\scripts\test.txt | Select-String "Failed" -quiet
Get-Content c:\config.sys |Select-String files
Get-Content d:\scripts\test.txt | Select-String "Failed" -quiet -casesensitive
```

H.Les redirections

On peut créer des fichiers avec les opérateurs de redirection usuels : > et >>

I.Création d'un fichier

La différence entre Out-File et Set-Content est que le premier ne sait créer que des fichiers texte.


```
Get-Process | Tee-Object -file d:\scripts\test.txt
```

J.Effacer le contenu d'un fichier

```
Clear-Content d:\scripts\test.txt
$A = Get-Date; Add-Content d:\test.log $A+`n
```

K.Convertir en Html

```
Get-Process | ConvertTo-Html | Set-Content d:\scripts\test.htm
Get-Process | ConvertTo-Html name,path,fileversion | Set-Content
d:\scripts\test.htm
Get-Process | ConvertTo-Html name,path,fileversion -title "Process Information" |
Set-Content d:\scripts\test.htm
Get-Process |
ConvertTo-Html name,path,fileversion -title "Process Information" -body
"Information about the processes running on the computer." |
Set-Content d:\scripts\test.htm
Get-Process |
ConvertTo-Html name,path,fileversion -title "Process Information" -body
"<H2>Information about the processes running on the computer.</H2>" |
Set-Content d:\scripts\test.htm
Get-ChildItem c:\windows\*.exe | ConvertTo-Html name, length| Set-Content
d:\index.html
```

1.Utiliser une page CSS

```
Get-Service|where Status -eq 'running'|ConvertTo-HTML -Property Name,DisplayName
`
-Title 'Liste des services' `
-Body '<h1>Services qui s''exécutent</h1>'|Out-file c:\powershell\services.html
Get-Service|where Status -eq 'running'|ConvertTo-HTML -Property Name,DisplayName
`
-Head '<title>Areuhhh</title><link rel="stylesheet" type="text/css"
href="style.css"/>' `
-Body '<h1>Services qui s''exécutent</h1>'|Out-file c:\powershell\services.html
```

L.Conversion en JSON

```
Get-Process|ConvertTo-JSON
'{ "Temps": "Lundi 2 septembre 2013 17:45" }' | ConvertFrom-Json | Get-Member
-Name Temps
```

M.Compter les lignes d'un fichier

```
Get-Content c:\config.sys | Measure-Object
Get-Content d:\scripts\test.txt | Select-Object -last 5
```

N.Lire un fichier CSV

```
Import-Csv d:\scripts\test.txt
Import-Csv d:\scripts\test.txt | Where-Object {$_.department -eq "Finance"}
Import-Csv d:\scripts\test.txt | Where-Object {$_.department -ne "Finance"}
Import-Csv d:\scripts\test.txt | Where-Object {$_.department -eq "Finance" -and
$_.title -eq "Accountant"}
Import-Csv d:\scripts\test.txt | Where-Object {$_.department -eq "Research" -or
$_.title -eq "Accountant"}
```

O.Les fichiers XML

```
Get-ChildItem d:\scripts | Export-Clixml d:\scripts\files.xml
```



```
$A = Import-Clixml d:\scripts\files.xml
$A | Sort-Object length
```

P.Export CSV

La différence entre ConvertTo-CSV et Export-CSV est que la conversion pour ConvertTo est réalisée en mémoire. Attention aux gros tableaux !

```
Get-Process | Export-Csv d:\scripts\test.txt
Get-Process | Export-Csv d:\scripts\test.txt -encoding "unicode"
#TYPE System.Diagnostics.Process
Get-Process | Export-Csv d:\scripts\test.txt -notype
Get-Process | Export-Csv d:\scripts\test.txt –force
```

Q.Sauvegarde d'un fichier

```
Set-Content d:\scripts\test.txt "This is a test"
Get-Process|Set-Content d:\test.txt
```

R.Export Xml

```
Get-Process | Export-Clixml d:\scripts\test.xml
```

S.Sauvegarder dans un fichier texte

Outfile permet de choisir l'encodage avec le paramètre –Encoding.

```
Get-Process | Out-File d:\scripts\test.txt
Get-Process | Out-File d:\scripts\test.txt -width 120
```

T.Interactif

Get-Service|Out-GridView

U.Export / Import CSV Tableaux et Tableaux associatifs

VII.Registre

A.Lecture d'une clé

```
Get-ChildItem -Path hkcu:\
```

B.Créer une clé

Push-Location

Set-Location HKCU:

Test-Path .\Software\dsfc

New-Item -Path .\Software -Name dsfc

Pop-Location

C.Créer une valeur

New-ItemProperty -path HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run -name "Notepad" -value "C:\WINDOWS\NOTEPAD.EXE" -type string

Suppression de clé

Remove-Item

Lecture / Ecriture

```
$val = Get-ItemProperty -Path
hklm:software\microsoft\windows\currentversion\policies\system -Name "EnableLUA"
if($val.EnableLUA -ne 0)
{
 set-itemproperty -Path hklm:software\microsoft\windows\currentversion\policies\system
-Name "EnableLUA" -value 0
}
```


VIII.Exécution distante

A.Présentation

Powershell utilise le RPC. Il s'appuie sur le service WinRM (Gestion à distance de Windows).
Au niveau du par-feu, vérifiez que les règles liées à la gestion distante soient activées.

Pour vérifier que le service s'exécute, tapez : netstat –ano|find "5985".
Pour configurer le service, tapez sous Powershell : Enable-PSRemoting. Vous disposez aussi de la commande winrm –quickconfig.
Pour vérifier la configuration : winrm get winrm/config

1.Sécurité

```
Enable-PSRemoting
Enter-PSSession –ComputerName host –Credential domain\user
Get-PSSessionConfiguration
```

B.Authentification

Dans un domaine, elle est de type Kerberos. Sinon, elle est en mode Negotiate (NTLM de poste à poste)

C.Machines de confiance (Poste à poste)

C'est du côté client.

```
Set-Item WSMan:\localhost\client\trustedhosts -value ACERARIEN -force –Concate-
nate
Get-Item WSMan:\localhost\client\trustedhosts
```

Pour verifier : winrm get winrm/config
Au niveau du registre, passez le paramètre
HKLM\SOFTWARE\Microsoft\windows\CurrentVersion\Policies\System\LocalAccountTokenFilterPolicy
En Powershell :

Set-ItempProperty –Path HKLM:\SOFTWARE\Microsoft\windows\CurrentVersion\Policies\System –name LocalAccountTokenFilterPolicy –Value 1 –Type DWord

D.Droits

Seuls les utilisateurs des groupes Administrateurs et Utilisateurs de gestion à distance peuvent se connecter via WinRM.

```
Set-PSSessionConfiguration -ShowSecurityDescriptorUI -Name Microsoft.PowerShell
```

E.Sessions

1.Session temporaire

Implicite par Invoke-Command et Enter-PSSession
Enter-PSSession –ComputerName ACERARIEN
Pour qu'elle soit permanente, ajoutez le paramètre – Session

2.Session permanente

New-PSSession –ComputerName ACERARIEN

3.Exécution distante

Invoke-Command –ComputerName ACERARIEN –ScriptBlock {$env::PATH}

4.Rappel de la session

$session=New-PSSession –ComputerName ACERARIEN
Invoke-Command –Session $session –ScriptBlock {$env::PATH}

F.Liste des commandes possibles

```
Get-Help * -Parameter ComputerName
```

G.Exemples

1.Invoke-Command

```
EXit-PSSession
$motdepasse = ConvertTo-SecureString "password" -AsPlainText -Force
$authentification = New-Object System.Management.Automation.PSCredential `
("MF230\Administrateur",$motdepasse)
$session=New-PSSession –ComputerName MF230 -Credential $authentification
$path=Invoke-Command -ScriptBlock {$env:computername} -Session $session
$cmd=Invoke-Command -ScriptBlock {&ipconfig} -Session $session
clear
$path
$cmd
```

2.Get-Process

```
$motdepasse = ConvertTo-SecureString "password" -AsPlainText -Force
$authentification = New-Object System.Management.Automation.PSCredential `
("MF230\Administrateur",$motdepasse)
Enter-PSSession –ComputerName MF230 -Credential $authentification
Get-Process -ComputerName MF230
```


IX.Modules Windows 8 et Windows 2012

A.NetAdapter

1.Importer le module NetAdapter

```
Import-Module NetAdapter
```

2.Profil

```
Get-NetConnectionProfile
```

3.Lister les périphériques réseaux

```
Get-NetAdapter
```

4.Elements attachés à la carte réseau

```
Get-NetAdapterBinding Ether*|Where-Object Enabled
```

5.Désactiver IPv6

```
Get-NetAdapterBinding -DisplayName *TCP/IPv6* | Disable-NetAdapterBinding
```

B.Partage réseau SmbShare

```
Import-Module SmbShare
Get-SmbShare
New-SmbShare –Path C:\test –Name test
Remove-SmbShare –Name test
Get-SmbSession
Get-SmbSession -ClientUserName *admin* | Close-SmbSession
Get-SmbShareAccess -Name test
Get-SmbShareAccess –Name test | Revoke-SmbShareAccess – AccountName Everyone
Block-SmbShareAccess –Name test –AccountName Everyone
Get-SMBOpenFile | Select-Object ClientComputerName,ClientUserName,Path
Get-SMBOpenFile | Select-Object ClientComputerName,ClientUserName,Path
Get-SmbOpenFile -ClientUserName mdn\administrator | Close-SmbOpenFile
```

C.Impression

```
Import-Module PrintManagement
Get-Printer -Name *Brother* | Select-Object Name,Type,DriverName,PortName
Get-Printer -Name *Brother* | Get-PrintJob | Remove-PrintJob
```

D.ODBC

```
Import-Module wdac
Get_OdbcDsn
Add-OdbcDsn -Name InternalDsn -DsnType User -DriverName "SQL Server"
-SetPropertyValue @("Database=LocalDatabase","Server=sql2008")
```

E.DNS

```
Resolve-DnsName –Name yahoo.fr | Format-List
Get-DnsClientCache| Select-Object -Property Name
Get-DNSClientServerAddress|Where-Object ServerAddresses
```


F.Disque

```
Import-Module Storage
Get-Disk
Get-Volume | Select-Object -Property DriveLetter,FileSystemLabel,Size
Initialize-Disk
New-Partition
Format-Volume -DriveLetter D|Format-List
```

G.Drivers

```
Get-WindowsDriver -Online | where date -gt 10/8/2012
```

Applications

```
Get-AppxPackage | Select Name, Version, Publisher | Where Publisher -Match
Microsoft | Sort Name
```


X.A tester

A.Panneau de configuration

```
Get-ControlPanelItem -Name Affichage
```

B.Renommer un ordinateur

```
Rename-Computer –ComputerName anciennom –NewName nouveaunom
–DomainCredential nouveaunom\administrateur –Force –Restart
```

C.Windows Core

```
Add-WindowsFeature Server-Gui-Shell, Server-Gui-Mgmt-Infra
Install-WindowsFeature Server-Gui-Shell, Server-Gui-Mgmt-Infra
```


XI.Active Directory

A.ADSI

Pour les versions antérieures à Windows 2008.
Il permet de gérer la base de comptes locaux.

1.Gestion des groupes locaux

a)Liste des groupes et des utilisateurs locaux

```
$conn=[ADSI]"WinNT://."
$conn.Children|Where SchemaClassName -eq 'group'|Select -ExpandProperty Name
$conn.Children|Where SchemaClassName -eq 'user'|Select -ExpandProperty Name
```

Membre d'un groupe

```
$conn=[ADSI]"WinNT://./Administrateurs,group"
$conn.Invoke('Members')|Foreach{
$_.GetType().InvokeMember('Name','GetProperty',$null,$_,$Null)
}
```

Ajout à un groupe

```
$conn=[ADSI]"WinNT://./Utilisateurs,group"
$conn.Add("WinNT://Administrateur")
```

b)Supprimer un membre d'un groupe

```
$conn=[ADSI]"WinNT://./Utilisateurs,group"
$conn.Remove("WinNT://Administrateur")
```

Lister les utilisateurs

```
$adsi = [ADSI]"WinNT://."
$adsi.psbase.children | where {$_.psbase.schemaClassName -match "user"} | select
@{n="Name";e={$_.name}}
```

Créer un groupe

```
$conn = [ADSI]"WinNT://."
$ogrp= $conn.Create('group','test')
$ogrp.Put('Description','Groupe de test')
$ogrp.SetInfo()
$ogrp.Dispose()
$conn.Dispose()
```

Renommer un groupe

```
$conn = [ADSI]"WinNT://./test,group"
$conn.PSBase.rename('test2')
$conn.setInfo()
$conn.Dispose()
```

Gestion des utilisateurs

c)Création d'un compte utilisateur

Les méthodes, propriétés utilisables sont indiquées dans mon support consacré à cette technologie sur mon site.

```
Clear
```



```
$oDom = [ADSI]"WinNT://."
$oUser=$oDom.Create("user","denis")
$oUser.PSBase.InvokeSet('Description','Big Boss')
$oUser.SetPassword("denis")
$oUser.SetInfo()
$oUser.Dispose()
$oDom.Dispose()
```

d)Modifier un compte local

```
Clear
$oUser = [ADSI]"WinNT://./denis,user"
$oUser.PSBase.InvokeSet('Description','Denis')
$oUser.SetInfo()
$oUser.Dispose()
```

e)Lister les propriétés d'un utilisateur

```
Clear
$oUser = [ADSI]"WinNT://./Administrateur,user"
$oUser.PSAdapted
$oUser.PSBase.InvokeGet('LastLogin').DateTime
$oUser.PSBase.InvokeGet('PasswordAge')
```

B.Module (à partir de Windows Server 2008)

1.Import

```
Import-Module ActiveDirectory
Get-Module ActiveDirectory
Get-command –Module ActiveDirectory
```

2.Liste des lecteurs

AD apparaît dans la liste des lecteurs !

```
Get-PSDrive
```

3.Gestion de l'annuaire

a)Lister l'annuaire

```
Get-ChildItem 'AD:\OU=Domain Controllers,DC=dutout,DC=net'
```

Requêtes

```
Get-ADObject -LDAPFilter '(&(objectCategory=person)(objectClass=user))'
Get-ADObject -LDAPFilter '(name=*acer*)'
```

b)filtres

```
Get-ADObject -Filter {objectClass -eq 'computer'}
```

Pour la liste des comptes désactivés :

```
Get-ADObject -Filter {(userAccountControl -eq 514) -and (objectClass -eq 'user')}
```

c)Vitesse d'interrogation

```
Measure-Command{Get-ADObject -Filter {(name -like '*admin*') -and (ObjectClass
-eq 'group')}}
Measure-Command{Get-ADObject -LDAPFilter '(name=*admin*)' | Where ObjectClass -eq
'group'}
Measure-Command{Get-ADObject -LDAPFilter '(name=*admin*)'}
Measure-Command{Get-ADObject -Filter {name -like '*admin*'}}
```


Lire les propriétés

```
Get-ItemProperty -Path 'AD:\CN=denis,OU=Informatique,OU=Services généraux,DC=du-
tout,DC=net' -name displayName

Get-ItemProperty -Path 'AD:\CN=denis,OU=Informatique,OU=Services généraux,DC=du-
tout,DC=net' -name displayName|Select-Object -ExpandProperty displayName
(Get-ItemProperty -Path 'AD:\CN=denis,OU=Informatique,OU=Services généraux,DC=du-
tout,DC=net' -name displayName).displayName
```

d)Modifier une propriété

```
$path='AD:\CN=denis,OU=Informatique,OU=Services généraux,DC=dutout,DC=net'
Set-ItemProperty -Path $path -name displayName -value 'Szalkowski Denis'
(Get-ItemProperty -Path $path -name displayName).displayName
```

e)Déplacement d'un objet

```
$old='AD:\OU=Informatique,DC=dutout,DC=net'
$new='AD:\OU=Services généraux,DC=dutout,DC=net'
Move-Item -Path $old -Destination $new
```

4.Les utilisateurs

a)Liste des utilisateurs

```
Get-ADUser -Filter * |Select name
Get-ADUser -Filter * |Select name
Get-ADUser -Filter * -Properties whenCreated|Select Name,whenCreated
Get-ADUser -Filter * -SearchBase 'OU=Informatique,OU=Services
généraux,DC=dutout,DC=net'|Select name
```

b)Création d'un utilisateur

```
$mdp=ConvertTo-SecureString 'paul' -AsPlainText -Force
New-ADUser -SamAccountName paul  -Name paul -Path 'OU=Informatique,OU=Services
généraux,DC=dutout,DC=net' -AccountPassword $mdp
```

c)Modifier un mot de passe

```
$mdp=ConvertTo-SecureString 'paul' -AsPlainText -Force
Set-ADAccountPassword -Identity paul -NewPassword $mdp -Reset
Set-ADUser -Identity paul -Enabled $true
```

Effacer un utilisateur

```
Remove-ADUser -identity:paul -Confirm:$false
```

lire les attributs

```
Get-ADUser -Identity denis -Properties *
Get-ADUser -Identity denis -Properties CN,displayName
```

Modifier des attributs

```
Set-ADUser -identity Denis -Replace @{
Description='Formateur Powershell';
TelephoneNumber='0670373191';
OtherTelephone=@('0232677952')}
```

effacer un attribut

```
Set-ADUser -identity denis -Clear OtherTelephone
```


Les groupes

Commandes relatives aux groups

Get-Command –Module ActiveDirectory –Name *group*

Liste des groupes

```
Get-AdGroup -Filter *|Select Name
Get-AdGroup -Filter {groupScope -eq 'DomainLocal'}|Select Name
Get-AdGroup -Filter * -SearchBase 'OU=Informatique,OU=Services généraux,DC=du-
tout,DC=net'
```

d)Création de groupes

```
New-ADGroup -Name Formateurs -GroupScope DomainLocal -GroupCategory Security
-Path 'OU=Informatique,OU=Services généraux,DC=dutout,DC=net'
```

e)Membres d'un groupe

```
Get-ADGroupMember -Identity Administrateurs|Select name
```

f)Ajout à un groupe

```
Add-ADGroupMember -Identity Administrateurs -Members denis,thierry
Add-ADPrincipalGroupMembership thierry -MemberOf Administrateurs
```

Supprimer les membres d'un groupe

Pour ces deux commandes, vous pouvez utilizer le paramètre –Confirm:$false
Remove-ADGroupMember
Remove-ADPrincipalGroupMemberShip

Suppression d'un groupe

Remove-ADGroup

C.Déploiement (2012)

```
Import-Module ADDSDeployment
```

1.Ajout de la forêt

```
Install-ADDSForest –DomainName dsfc.local –DomainMode Win2008R2 –ForestMode
Win2008R2 –RebootOnCompletion
```

2.Ajout du DC

```
Install-ADDSDomainController –DomainName dsfc.local
```

3.Désinstallation du DC

```
Uninstall-ADDSDomainController -LastDomainControllerInDomain
-RemoveApplicationPartitions
```


XII.PowerShell sous Windows 2008 R2

A.Source

http://technet.microsoft.com/fr-fr/library/dd378843%28WS.10%29.aspx

B.La listes des cmdlets

Cmdlet	Description
Add-ADComputerServiceAccount	Adds one or more service accounts to an Active Directory computer.
Add-ADDomainControllerPasswordReplicationPolicy	Adds users, computers, and groups to the Allowed List or the Denied List of the read-only domain controller (RODC) Password Replication Policy (PRP).
Add-ADFineGrainedPasswordPolicySubject	Applies a fine-grained password policy to one more users and groups.
Add-ADGroupMember	Adds one or more members to an Active Directory group.
Add-ADPrincipalGroupMembership	Adds a member to one or more Active Directory groups.
Clear-ADAccountExpiration	Clears the expiration date for an Active Directory account.
Disable-ADAccount	Disables an Active Directory account.
Disable-ADOptionalFeature	Disables an Active Directory optional feature.
Enable-ADAccount	Enables an Active Directory account.
Enable-ADOptionalFeature	Enables an Active Directory optional feature.
Get-ADAccountAuthorizationGroup	Gets the Active Directory security groups that contain an account.
Get-ADAccountResultantPasswordReplicationPolicy	Gets the resultant password replication policy for an Active Directory account.
Get-ADComputer	Gets one or more Active Directory computers.
Get-ADComputerServiceAccount	Gets the service accounts that are hosted by an Active Directory computer.
Get-ADDefaultDomainPasswordPolicy	Gets the default password policy for an Active Directory domain.
Get-ADDomain	Gets an Active Directory domain.
Get-ADDomainController	Gets one or more Active Directory domain controllers,

	based on discoverable services criteria, search parameters, or by providing a domain controller identifier, such as the NetBIOS name.
Get-ADDomainControllerPasswordReplicationPolicy	Gets the members of the Allowed List or the Denied List of the RODC PRP.
Get-ADDomainControllerPasswordReplicationPolicyUsage	Gets the resultant password policy of the specified ADAccount on the specified RODC.
Get-ADFineGrainedPasswordPolicy	Gets one or more Active Directory fine-grained password policies.
Get-ADFineGrainedPasswordPolicySubject	Gets the users and groups to which a fine-grained password policy is applied.
Get-ADForest	Gets an Active Directory forest.
Get-ADGroup	Gets one or more Active Directory groups.
Get-ADGroupMember	Gets the members of an Active Directory group.
Get-ADObject	Gets one or more Active Directory objects.
Get-ADOptionalFeature	Gets one or more Active Directory optional features.
Get-ADOrganizationalUnit	Gets one or more Active Directory OUs.
Get-ADPrincipalGroupMembership	Gets the Active Directory groups that have a specified user, computer, or group.
Get-ADRootDSE	Gets the root of a domain controller information tree.
Get-ADServiceAccount	Gets one or more Active Directory service accounts.
Get-ADUser	Gets one or more Active Directory users.
Get-ADUserResultantPasswordPolicy	Gets the resultant password policy for a user.
Install-ADServiceAccount	Installs an Active Directory service account on a computer.
Move-ADDirectoryServer	Moves a domain controller in AD DS to a new site.
Move-ADDirectoryServerOperationMasterRole	Moves operation master (also known as flexible single master operations or FSMO) roles to an Active Directory domain controller.
Move-ADObject	Moves an Active Directory object or a container of objects to a different container or domain.
New-ADComputer	Creates a new Active Directory computer.
New-ADFineGrainedPasswordPolicy	Creates a new Active Directory fine-grained password policy.

New-ADGroup	Creates an Active Directory group.
New-ADObject	Creates an Active Directory object.
New-ADOrganizationalUnit	Creates a new Active Directory OU.
New-ADServiceAccount	Creates a new Active Directory service account.
New-ADUser	Creates a new Active Directory user.
Remove-ADComputer	Removes an Active Directory computer.
Remove-ADComputerServiceAccount	Removes one or more service accounts from a computer.
Remove-ADDomainControllerPasswordReplicationPolicy	Removes users, computers, and groups from the Allowed List or the Denied List of the RODC PRP.
Remove-ADFineGrainedPasswordPolicy	Removes an Active Directory fine-grained password policy.
Remove-ADFineGrainedPasswordPolicySubject	Removes one or more users from a fine-grained password policy.
Remove-ADGroup	Removes an Active Directory group.
Remove-ADGroupMember	Removes one or more members from an Active Directory group.
Remove-ADObject	Removes an Active Directory object.
Remove-ADOrganizationalUnit	Removes an Active Directory OU.
Remove-ADPrincipalGroupMembership	Removes a member from one or more Active Directory groups.
Remove-ADServiceAccount	Removes an Active Directory service account.
Remove-ADUser	Removes an Active Directory user.
Rename-ADObject	Changes the name of an Active Directory object.
Reset-ADServiceAccountPassword	Resets the service account password for a computer.
Restore-ADObject	Restores an Active Directory object.
Search-ADAccount	Gets Active Directory user, computer, and service accounts.
Set-ADAccountControl	Modifies user account control (UAC) values for an Active Directory account.
Set-ADAccountExpiration	Sets the expiration date for an Active Directory account.

Set-ADAccountPassword	Modifies the password of an Active Directory account.
Set-ADComputer	Modifies an Active Directory computer.
Set-ADDefaultDomainPasswordPolicy	Modifies the default password policy for an Active Directory domain.
Set-ADDomain	Modifies an Active Directory domain.
Set-ADDomainMode	Sets the domain functional level for an Active Directory domain.
Set-ADFineGrainedPasswordPolicy	Modifies an Active Directory fine-grained password policy.
Set-ADForest	Modifies an Active Directory forest.
Set-ADForestMode	Sets the forest mode for an Active Directory forest.
Set-ADGroup	Modifies an Active Directory group.
Set-ADObject	Modifies an Active Directory object.
Set-ADOrganizationalUnit	Modifies an Active Directory OU.
Set-ADServiceAccount	Modifies an Active Directory service account.
Set-ADUser	Modifies an Active Directory user.
Uninstall-ADServiceAccount	Uninstalls an Active Directory service account from a computer.
Unlock-ADAccount	Unlocks an Active Directory account.

C.La gestion des utilisateurs

Get-ADUser *UserName* *Liste les informations relatives à un nouvel utilisateur*

Get-ADUser -Filter {Name -like "**SearchVariables***"} Filtrage des informations
Get-ADUser -Filter {Name -like "*minis

New-ADUser –Name "*FirstName LastName*" –SamAccountName "*firstname.lastname*" –Description "*Description*" –Department "*Department*" –Office "*Office Location*" –Path "cn=users,dc=*domain*,dc=*domain*" –Enabled $true (For example if we were to create one of the users via this method instead of the import csv method below we would enter the following command: New-ADUser –Name "Ben Jones" –SamAccountName "ben.jones" –Description "Managing Directory" –Department "Sales" –Office "Sydney" –Path "ou=users,ou=sydney,dc=windowslab,dc=local" –Enabled $true)

Import-CSV C:\users.csv | New-ADUser (The Import-CSV cmdlet will import a list of users from a CSV file created in excel. Below is a sample of one I have created. You can create more columns using the same Active Directory User Account settings.)

Remove-ADUser *UserName* (Removes/Deletes an AD User. You will be asked if you are sure you want to perform this action. You can also use a filter similar to the Get-ADUser command above)

Set-ADUser *ADUser* -*Variable* (This command will set the user fields for the specified user account. For example if I want to specify or change the Office location field for a specific user I would type: Set-ADUser ben.jones -Office Brisbane. This command can also be used with the Filter command for mass additions or changes.)

D.Les groupes

Get-ADGroup *GroupName* (Lists information about a specific Group. If the Group contains spaces don't forget to use "")

Get-ADGroup -Filter {Name -like "**SearchVariables***"} For Example Get-ADGroup -Filter {Name -like "*mins*"} to search for all Groups containing the word mins i.e. Domain Ad**mins**, etc

New-ADGroup -name *GroupName* -GroupScope *Global*|*Universa*l -Description *"Description"* -DisplayName *DisplayName* -SamAccountName *AccountName* (For example to create a Global Group call TestGroup I would use the following syntax
New-ADGroup -name TestGroup -GroupScope Global -Description "New Group Test" -DisplayName TestGroup -SamAccountName TestGroup

Remove-ADGroup *GroupName* (Removes/Deletes an ADGroup. You will be asked if you are sure you want to perform this action. You can also use a filter similar to the Get-ADGroup command above)

Set-ADGroup *GroupName* -*Variable* (This command will set the definable fields for the specified Group account. For example if I want to specify or change the Description field for a specific group I would type: Set-ADGroup TestGroup -Description "Demo Group". This command can also be used with the Filter command for mass additions or changes.)

XIII.Quelques exemples

A.Liste des fichiers exécutés sur la machine

Ce script a pour objet de lireles fichiers qui ont été exécutés au moins une fois sur la machine. Cette liste associée au mécanisme du *Prefetcher* se situe dans le dossier *c:\windows\prefetch* de votre disque dur.

```
$rows=Get-ChildItem c:\windows\prefetch |Where-Object {$_.Name -match '\.EXE'}|
Select-Object Name
Foreach($row in $rows)
{
    $i = $row.Name.IndexOf(".")
    $a = $row.Name.substring(0,$i+4)
    Write-Host $a
}
```

B.Liste des services à partir du registre

```
Clear
$keys=Get-ChildItem hklm:SYSTEM\CurrentControlSet\services|Select-Object Name
$t = "boot","system","auto","manual"
Foreach($key in $keys)
{
    $a=$key.Name.Replace("HKEY_LOCAL_MACHINE\","hklm:")
    $s=(Get-ItemProperty $a).Start
    If($s -lt 4 -and $s -ge 0)
    {
        $p=$a.LastIndexOf('\')+1
        $l=$a.Length
        Write-Host $t[$s] `t $a.SubString($p,$l-$p)
        #
    }
}
```

C.Utilisation des composants WSH Windows Scripting Host

L'intérêt du PowerShell est de vous permettre d'employer les objets associés à la technologie Windows Scripting Host. : Wscript.NetWork et Wscript.Shell. Vous les retrouverez dans mon support consacré à cette technologie sur mon site.

1.Wscript.Shell

```
$oShell = New-Object -com Wscript.Shell
$oShell.Run("c:\windows\system32\calc.exe")
Pour disposer de toutes les methods :
$oShell|Get-Member
```

2.Wscript.Network

```
$oNetwork = New-Object -com Wscript.Network
#$oNetWork.UserName
#$env:USERNAME
#$oNetwork.ComputerName
Try
{
    $oNetwork.RemoveNetworkDrive('P:')
}
Catch
{
```



```
    'Ca marche pas'
}
Finally
{
    $oNetwork.MapNetworkDrive('P:', '\\10.114.3.152\PatchWin7',$false,`
    'MF231\Administrateur','password')
}
$oNetwork=$null
    Get-ChildItem x:\
}
$oNetWork.Dispose
```

3.Partage d'imprimante

```
$Path = "\\10.114.3.153\hpjpp"
$oNw = New-Object -com Wscript.Network
Try
{
    $oNw.RemoveWindowsPrinterConnection($path)
}
Catch
{
}
Finally
{
    $oNw.AddWindowsPrinterConnection($path)
}
```

4.Scripting.FileSystemObject

```
$oFso = New-Object -com Scripting.FileSystemObject
$oFile=$oFso.GetFile("c:\config.sys")
Write-Host $oFile.DateLastAccessed
```

D.MySQL : lecture de tables

```
[void][system.reflection.Assembly]::LoadFrom("C:\Program Files\MySQL\MySQL
Connector Net 6.3.6\Assemblies\v2.0\MySql.Data.dll")
Cls
$strConn="DataSource=localhost;Database='veille';User ID='root';Password=''"
Try
{
    $oConn = New-Object MySql.Data.MySqlClient.MySqlConnection
    $oConn.ConnectionString = $strConn
    $oConn.Open()
    #$oConn  = New-Object MySql.Data.MySqlClient.MySqlConnection($strConn)
}
Catch [System.Exception]
{
    $e = $_.Exception
    Write-Host $e.Message
}
Finally
{
}
$oSql = New-Object MySql.Data.MySqlClient.MySqlCommand
$oSql.Connection = $oConn
$oSql.CommandText = "SELECT * from moteur"
$oReader = $oSql.ExecuteReader()
while($oReader.Read())
{
#    Write-Host $oReader.GetString('moteur_url')
```

```
    for ($i= 0; $i -lt $oReader.FieldCount; $i++)
    {
        Write-Host $oReader.GetValue($i).ToString()
    }
}
$oReader.Close()
$oReader.Dispose()
$oAdapter = New-Object MySql.Data.MySqlClient.MySqlDataAdapter($oSql)
$oDataSet = New-Object System.Data.DataSet
$oAdapter.Fill($oDataSet,"data")
$data = $oDataSet.Tables["data"]
$data | Format-Table
$data.Dispose()
$oDataSet.Dispose()
$oAdapter.Dispose()
$oSql.Dispose()
$oConn.Close()
$oConn.Dispose()
#    $sql = New-Object MySql.Data.MySqlClient.MySqlCommand
#    $sql.Connection = $oConn
#    $sql.CommandText = "INSERT INTO computer_details (computer_id, mac, dhcp,
model, domain, manufacturer, type, memory, ip, servicetag, lastimagedate,
servicepack, os, biosrev, scriptversion, lastrun, ou) VALUES ('$resultID',
'$macAddress', '$dhcp', '$model', '$domain', '$manufacturer', '$systemType',
'$memory', '$ipAddress', '$servicetag', NOW(), '$servicePack',
'$operatingSystem', '$biosrev', '$version', NOW(), '$ou' )"
#    $sql.ExecuteNonQuery()
#    $dbconnect.Close()
```

E.Les compteurs

```
do
{
    (Get-Counter -Counter '\Interface réseau(*)\Octets reçus/s').CounterSamples|
where InstanceName -like 'broadcom*'|Select CookedValue
}
while($true)
```

F.MySQL : inventaire

1.La table

```
CREATE TABLE `logiciel` (
  `logiciel_nom` varchar(255) DEFAULT NULL,
  `logiciel_machine` varchar(15) DEFAULT NULL,
  `logiciel_date` varchar(20) DEFAULT NULL,
  UNIQUE KEY `uk_logiciel` (`logiciel_nom`,`logiciel_machine`)
)
```

2.Le script

```
Clear
[void][system.reflection.Assembly]::LoadFrom("C:\Program Files\MySQL\MySQL
Connector Net 6.3.6\Assemblies\v2.0\MySql.Data.dll")
$strConn="DataSource=localhost;Database='inventaire';User ID='root';Password='"
$oConn = New-Object MySql.Data.MySqlClient.MySqlConnection
$oConn.ConnectionString = $strConn
Try
{
```

```
    $oConn.Open()
}
Catch [System.Exception]
{
    $e = $_.Exception
    Write-Host $e.Message
}
$req = New-Object MySql.Data.MySqlClient.MySqlCommand
$req.Connection=$oConn
$content=Get-ChildItem c:\windows\prefetch\*.pf
$oNetwork = New-Object -com Wscript.Network
$c=$oNetwork.ComputerName
ForEach($row in $content)
{
    $n=$row.Name
    $d=[datetime](Get-Item $row).LastAccessTime
    $p=$n.LastIndexOf('-')
    $s=$n.SubString(0,$p)
    $sql="INSERT INTO logiciel VALUES('"+$s+"','"+$c+"','"+$d+"')"
    $req.CommandText = $sql
    Try
    {
        $req.ExecuteNonQuery()
    }
    Catch
    {
        $sql="UPDATE logiciel SET logiciel_date='"+$d+"'
        WHERE logiciel_nom='"+$s+"' AND logiciel_machine='"+$c+"'"
        $req.CommandText = $sql
        $req.ExecuteNonQuery()
    }
}
$req.Dispose()
$oConn.Close()
$oConn.Dispose()
```


XIV.Quelques sites

PowerShell 3.0 est en passe de s'imposer comme technologie de scripting dans les environnements Windows. Derrière une simplicité apparente, se cache parfois une réelle complexité. Ces quelques liens vous permettront, je l'espère, de progresser dans un langage qui s'appuie sur le Framework .Net 4.0.

A.Sites en français

- Windows PowerShell (**site officiel**) : guide
- Centre de scripts Windows PowerShell (**site officiel**) : téléchargements, scripts, mémento
- Galerie de scripts PowerShell (**site officiel**) : téléchargements, scripts
- Laurent Dardenne : liens, tutoriaux
- PowerShell-Scripting.com : articles, tutoriaux, scripts, mémento
- via PowerShell : tutoriaux, liens
- SysKB : scripts

B.Sites en anglais

- CodePlex (modules PowerShell Open Source) : téléchargements
- PowerShell.com : scripts, tutoriaux
- Sapien Technologies : téléchargements, scripts
- Precision Computing : scripts

C.Téléchargements

- Microsoft Framework .Net 4.0 **(site officiel)**
- Windows Management Framework 3.0 **(site officiel)**
- PowerShellPack **(site officiel)**
- Outils d'administration de serveur distant pour Windows 7 SP1 (**site officiel**)
- PowerShell Scriptomatic (en)

D.Éditeurs gratuits

- PowerGUI
- PowerShell Plus

XV.Annexe 1 : cmdlets et fonctions présentes sous Windows Server 2012

A.Les CmdLets

Add-AppxPackage
Add-AppxProvisionedPackage
Add-BitsFile
Add-CertificateEnrollmentPolicyServer
Add-ClusteriSCSITargetServerRole
Add-Computer
Add-Content
Add-History
Add-IscsiVirtualDiskTargetMapping
Add-JobTrigger
Add-KdsRootKey
Add-Member
Add-PSSnapin
Add-RoleMember
Add-SqlAvailabilityDatabase
Add-SqlAvailabilityGroupListenerStaticIp
Add-Type
Add-WindowsDriver
Add-WindowsPackage
Backup-ASDatabase
Backup-SqlDatabase
Checkpoint-Computer
Checkpoint-IscsiVirtualDisk
Clear-Content
Clear-EventLog
Clear-History
Clear-Item
Clear-ItemProperty
Clear-KdsCache
Clear-Tpm
Clear-Variable
Clear-WindowsCorruptMountPoint
Compare-Object
Complete-BitsTransfer
Complete-DtcDiagnosticTransaction
Complete-Transaction
Confirm-SecureBootUEFI
Connect-PSSession
Connect-WSMan
ConvertFrom-Csv
ConvertFrom-Json
ConvertFrom-SecureString
ConvertFrom-StringData
Convert-IscsiVirtualDisk
Convert-Path
ConvertTo-Csv
ConvertTo-Html
ConvertTo-Json
ConvertTo-SecureString
ConvertTo-TpmOwnerAuth
ConvertTo-Xml
Convert-UrnToPath
Copy-Item
Copy-ItemProperty
Debug-Process
Decode-SqlName
Disable-ComputerRestore
Disable-JobTrigger
Disable-PSBreakpoint
Disable-PSRemoting
Disable-PSSessionConfiguration
Disable-ScheduledJob
Disable-SqlAlwaysOn
Disable-TpmAutoProvisioning
Disable-WindowsErrorReporting
Disable-WindowsOptionalFeature
Disable-WSManCredSSP
Disconnect-PSSession
Disconnect-WSMan
Dismount-IscsiVirtualDiskSnapshot
Dismount-WindowsImage
Enable-ComputerRestore
Enable-JobTrigger
Enable-PSBreakpoint
Enable-PSRemoting
Enable-PSSessionConfiguration
Enable-ScheduledJob
Enable-SqlAlwaysOn
Enable-TpmAutoProvisioning
Enable-WindowsErrorReporting
Enable-WindowsOptionalFeature
Enable-WSManCredSSP
Encode-SqlName
Enter-PSSession
Exit-PSSession
Expand-IscsiVirtualDisk
Export-Alias
Export-Certificate
Export-Clixml
Export-Console
Export-Counter
Export-Csv
Export-FormatData
Export-IscsiVirtualDiskSnapshot
Export-ModuleMember
Export-PfxCertificate
Export-PSSession
ForEach-Object
Format-Custom
Format-List
Format-SecureBootUEFI
Format-Table
Format-Wide
Get-Acl
Get-Alias
Get-AppLockerFileInformation
Get-AppLockerPolicy
Get-AppxPackage
Get-AppxPackageManifest
Get-AppxProvisionedPackage
Get-AuthenticodeSignature
Get-BitsTransfer
Get-BpaModel
Get-BpaResult
Get-Certificate
Get-CertificateAutoEnrollmentPolicy
Get-CertificateEnrollmentPolicyServer
Get-CertificateNotificationTask
Get-ChildItem
Get-CimAssociatedInstance
Get-CimClass
Get-CimInstance
Get-CimSession
Get-Command
Get-ComputerRestorePoint
Get-Content
Get-ControlPanelItem
Get-Counter
Get-Credential

```
Get-Culture
Get-DAPolicyChange
Get-Date
Get-Event
Get-EventLog
Get-EventSubscriber
Get-ExecutionPolicy
Get-FormatData
Get-Help
Get-History
Get-Host
Get-HotFix
Get-IscsiServerTarget
Get-IscsiTargetServerSet-
ting
Get-IscsiVirtualDisk
Get-IscsiVirtualDiskSnap-
shot
Get-Item
Get-ItemProperty
Get-Job
Get-JobTrigger
Get-KdsConfiguration
Get-KdsRootKey
Get-Location
Get-Member
Get-Module
Get-NfsMappedIdentity
Get-NfsNetgroup
Get-PfxCertificate
Get-PfxData
Get-Process
Get-PSBreakpoint
Get-PSCallStack
Get-PSDrive
Get-PSProvider
Get-PSSession
Get-PSSessionConfigura-
tion
Get-PSSnapin
Get-Random
Get-ScheduledJob
Get-ScheduledJobOption
Get-SecureBootPolicy
Get-SecureBootUEFI
Get-Service
Get-Tpm
Get-TraceSource
Get-Transaction
Get-TroubleshootingPack
Get-TypeData
Get-UICulture
Get-Unique
Get-Variable
Get-WheaMemoryPolicy
Get-WinAcceptLanguage-
FromLanguageListOptOut
Get-WinCultureFromLan-
guageListOptOut
Get-WinDefaultInputMe-
thodOverride
Get-WindowsDriver
Get-WindowsEdition
Get-WindowsErrorReporting
Get-WindowsImage
Get-WindowsOptionalFea-
ture
Get-WindowsPackage
Get-WinEvent
Get-WinHomeLocation
Get-WinLanguageBarOption
Get-WinSystemLocale
Get-WinUILanguageOverride
Get-WinUserLanguageList
Get-WmiObject
Get-WSManCredSSP
Get-WSManInstance
Group-Object
Import-Alias
Import-Certificate
Import-Clixml
Import-Counter
Import-Csv
Import-IscsiVirtualDisk
Import-LocalizedData
Import-Module
Import-PfxCertificate
Import-PSSession
Import-TpmOwnerAuth
Initialize-Tpm
Install-NfsMappingStore
Invoke-ASCmd
Invoke-BpaModel
Invoke-CimMethod
Invoke-Command
Invoke-Expression
Invoke-History
Invoke-Item
Invoke-PolicyEvaluation
Invoke-ProcessCube
Invoke-ProcessDimension
Invoke-ProcessPartition
Invoke-RestMethod
Invoke-Sqlcmd
Invoke-Troubleshooting-
Pack
Invoke-WebRequest
Invoke-WmiMethod
Invoke-WSManAction
Join-DtcDiagnosticResour-
ceManager
Join-Path
Join-SqlAvailabilityGroup
Limit-EventLog
Measure-Command
Measure-Object
Merge-Partition
Mount-IscsiVirtualDiskS-
napshot
Mount-WindowsImage
Move-Item
Move-ItemProperty
New-Alias
New-AppLockerPolicy
New-CertificateNotifica-
tionTask
New-CimInstance
New-CimSession
New-CimSessionOption
New-DtcDiagnosticTransac-
tion
New-Event
New-EventLog
New-IscsiServerTarget
New-IscsiVirtualDisk
New-Item
New-ItemProperty
New-JobTrigger
New-Module
New-ModuleManifest
New-NetIPsecAuthProposal
New-NetIPsecMainMode-
CryptoProposal
New-NetIPsecQuickMode-
CryptoProposal
New-NfsMappedIdentity
New-NfsNetgroup
New-Object
New-PSDrive
New-PSSession
New-PSSessionConfigura-
tionFile
New-PSSessionOption
New-PSTransportOption
New-PSWorkflowExecutio-
nOption
New-RestoreFolder
New-RestoreLocation
New-ScheduledJobOption
New-SelfSignedCertificate
New-Service
New-SqlAvailabilityGroup
New-SqlAvailabilityGrou-
pListener
New-SqlAvailabilityRepli-
ca
New-SqlHADREndpoint
New-TimeSpan
New-Variable
New-WebServiceProxy
New-WinEvent
New-WinUserLanguageList
New-WSManInstance
```


New-WSManSessionOption
Out-Default
Out-File
Out-GridView
Out-Host
Out-Null
Out-Printer
Out-String
Pop-Location
Push-Location
Read-Host
Receive-DtcDiagnosticTransaction
Receive-Job
Receive-PSSession
Register-CimIndicationEvent
Register-EngineEvent
Register-ObjectEvent
Register-PSSessionConfiguration
Register-ScheduledJob
Register-WmiEvent
Remove-AppxPackage
Remove-AppxProvisionedPackage
Remove-BitsTransfer
Remove-CertificateEnrollmentPolicyServer
Remove-CertificateNotificationTask
Remove-CimInstance
Remove-CimSession
Remove-Computer
Remove-Event
Remove-EventLog
Remove-IscsiServerTarget
Remove-IscsiVirtualDisk
Remove-IscsiVirtualDiskSnapshot
Remove-IscsiVirtualDiskTargetMapping
Remove-Item
Remove-ItemProperty
Remove-Job
Remove-JobTrigger
Remove-Module
Remove-NfsMappedIdentity
Remove-NfsNetgroup
Remove-PSBreakpoint
Remove-PSDrive
Remove-PSSession
Remove-PSSnapin
Remove-RoleMember
Remove-SqlAvailabilityDatabase
Remove-SqlAvailabilityGroup
Remove-SqlAvailabilityReplica
Remove-TypeData
Remove-Variable
Remove-WindowsDriver
Remove-WindowsPackage
Remove-WmiObject
Remove-WSManInstance
Rename-Computer
Rename-Item
Rename-ItemProperty
Repair-WindowsImage
Reset-ComputerMachinePassword
Resolve-DnsName
Resolve-Path
Restart-Computer
Restart-Service
Restore-ASDatabase
Restore-Computer
Restore-IscsiVirtualDisk
Restore-SqlDatabase
Resume-BitsTransfer
Resume-Job
Resume-Service
Resume-SqlAvailabilityDatabase
Save-Help
Save-WindowsImage
Select-Object
Select-String
Select-Xml
Send-DtcDiagnosticTransaction
Send-MailMessage
Set-Acl
Set-Alias
Set-AppLockerPolicy
Set-AuthenticodeSignature
Set-BitsTransfer
Set-BpaResult
Set-CertificateAutoEnrollmentPolicy
Set-CimInstance
Set-Content
Set-Culture
Set-Date
Set-ExecutionPolicy
Set-IscsiServerTarget
Set-IscsiTargetServerSetting
Set-IscsiVirtualDisk
Set-IscsiVirtualDiskSnapshot
Set-Item
Set-ItemProperty
Set-JobTrigger
Set-KdsConfiguration
Set-Location
Set-NfsMappedIdentity
Set-NfsNetgroup
Set-PSBreakpoint
Set-PSDebug
Set-PSSessionConfiguration
Set-ScheduledJob
Set-ScheduledJobOption
Set-SecureBootUEFI
Set-Service
Set-SqlAvailabilityGroup
Set-SqlAvailabilityGroupListener
Set-SqlAvailabilityReplica
Set-SqlHADREndpoint
Set-StrictMode
Set-TpmOwnerAuth
Set-TraceSource
Set-Variable
Set-WheaMemoryPolicy
Set-WinAcceptLanguageFromLanguageListOptOut
Set-WinCultureFromLanguageListOptOut
Set-WinDefaultInputMethodOverride
Set-WindowsEdition
Set-WindowsProductKey
Set-WinHomeLocation
Set-WinLanguageBarOption
Set-WinSystemLocale
Set-WinUILanguageOverride
Set-WinUserLanguageList
Set-WmiInstance
Set-WSManInstance
Set-WSManQuickConfig
Show-Command
Show-ControlPanelItem
Show-EventLog
Sort-Object
Split-Path
Start-BitsTransfer
Start-DtcDiagnosticResourceManager
Start-Job
Start-Process
Start-Service
Start-Sleep
Start-Transaction
Start-Transcript
Stop-Computer
Stop-DtcDiagnosticResourceManager
Stop-Job
Stop-Process
Stop-Service


```
Stop-Transcript
Suspend-BitsTransfer
Suspend-Job
Suspend-Service
Suspend-SqlAvailability-
Database
Switch-Certificate
Switch-SqlAvailability-
Group
Tee-Object
Test-AppLockerPolicy
Test-Certificate
Test-ComputerSecureChan-
nel
Test-Connection
Test-KdsRootKey
Test-ModuleManifest
Test-NfsMappedIdentity
Test-Path
Test-PSSessionConfigura-
tionFile
Test-SqlAvailabilityGroup
Test-SqlAvailabilityRe-
plica
Test-SqlDatabaseReplicaS-
tate
Test-WSMan
Trace-Command
Unblock-File
Unblock-Tpm
Undo-DtcDiagnosticTran-
saction
Undo-Transaction
Unregister-Event
Unregister-PSSessionCon-
figuration
Unregister-ScheduledJob
Update-FormatData
Update-Help
Update-List
Update-TypeData
Use-Transaction
Use-WindowsUnattend
Wait-Event
Wait-Job
Wait-Process
Where-Object
Write-Debug
Write-Error
Write-EventLog
Write-Host
Write-Output
Write-Progress
Write-Verbose
Write-Warning
```

B.Les fonctions

```
A:
Add-BCDataCacheExtension
Add-BitLockerKeyProtector
Add-DnsClientNrptRule
Add-DtcClusterTMMapping
Add-InitiatorIdToMasking-
Set
Add-NetIPHttpsCertBinding
Add-NetLbfoTeamMember
Add-NetLbfoTeamNic
Add-NetSwitchTeamMember
Add-OdbcDsn
Add-PartitionAccessPath
Add-PhysicalDisk
Add-Printer
Add-PrinterDriver
Add-PrinterPort
Add-RDServer
Add-RDSessionHost
Add-RDVirtualDesktopTo-
Collection
Add-TargetPortToMasking-
Set
Add-VirtualDiskToMasking-
Set
Add-VpnConnection
B:
Backup-BitLockerKeyPro-
tector
Block-SmbShareAccess
C:
cd..
cd\
Clear-BCCache
Clear-BitLockerAutoUnlock
Clear-Disk
Clear-DnsClientCache
Clear-Host
Close-SmbOpenFile
Close-SmbSession
Connect-IscsiTarget
Connect-VirtualDisk
Copy-NetFirewallRule
Copy-NetIPsecMainMode-
CryptoSet
Copy-NetIPsecMainModeRule
Copy-NetIPsecPhase1Auth-
Set
Copy-NetIPsecPhase2Auth-
Set
Copy-NetIPsecQuickMode-
CryptoSet
Copy-NetIPsecRule
D:
Disable-BC
Disable-BCDowngrading
Disable-BCServeOnBattery
Disable-BitLocker
Disable-BitLockerAutoUn-
lock
Disable-DAManualEntry-
PointSelection
Disable-MMAgent
Disable-NetAdapter
Disable-NetAdapterBinding
Disable-NetAdapterCheck-
sumOffload
Disable-NetAdapterEncap-
sulatedPacketTaskOffload
Disable-NetAdapterIPse-
cOffload
Disable-NetAdapterLso
Disable-NetAdapterPower-
Management
Disable-NetAdapterQos
Disable-NetAdapterRdma
Disable-NetAdapterRsc
Disable-NetAdapterRss
Disable-NetAdapterSriov
Disable-NetAdapterVmq
Disable-NetDnsTransition-
Configuration
Disable-NetFirewallRule
Disable-NetIPHttpsProfile
Disable-NetIPsecMainMode-
Rule
Disable-NetIPsecRule
Disable-NetNatTransition-
Configuration
Disable-OdbcPerfCounter
Disable-PhysicalDiskIndi-
cation
Disable-PSTrace
Disable-PSWSManCombined-
Trace
Disable-RDVirtualDeskto-
pADMachineAccountReuse
Disable-ScheduledTask
Disable-ServerManager-
StandardUserRemoting
Disable-Ual
Disable-WdacBidTrace
Disable-WSManTrace
Disconnect-IscsiTarget
Disconnect-NfsSession
Disconnect-RDUser
Disconnect-VirtualDisk
Dismount-DiskImage
```

```
E:
Enable-BCDistributed
Enable-BCDowngrading
Enable-BCHostedClient
Enable-BCHostedServer
Enable-BCLocal
Enable-BCServeOnBattery
Enable-BitLocker
Enable-BitLockerAutoUn-
lock
Enable-DAManualEntry-
PointSelection
Enable-MMAgent
Enable-NetAdapter
Enable-NetAdapterBinding
Enable-NetAdapterChecksu-
mOffload
Enable-NetAdapterEncapsu-
latedPacketTaskOffload
Enable-NetAdapterIPsecOf-
fload
Enable-NetAdapterLso
Enable-NetAdapterPowerMa-
nagement
Enable-NetAdapterQos
Enable-NetAdapterRdma
Enable-NetAdapterRsc
Enable-NetAdapterRss
Enable-NetAdapterSriov
Enable-NetAdapterVmq
Enable-NetDnsTransition-
Configuration
Enable-NetFirewallRule
Enable-NetIPHttpsProfile
Enable-NetIPsecMainMode-
Rule
Enable-NetIPsecRule
Enable-NetNatTransition-
Configuration
Enable-OdbcPerfCounter
Enable-PhysicalDiskIndi-
cation
Enable-PSTrace
Enable-PSWSManCombined-
Trace
Enable-RDVirtualDeskto-
pADMachineAccountReuse
Enable-ScheduledTask
Enable-ServerManagerStan-
dardUserRemoting
Enable-Ual
Enable-WdacBidTrace
Enable-WSManTrace
Export-BCCachePackage
Export-BCSecretKey
Export-RDPersonalVirtual-
DesktopAssignment
Export-ScheduledTask
F:
Format-Volume
G:
Get-AppxLastError
Get-AppxLog
Get-BCClientConfiguration
Get-BCContentServerConfi-
guration
Get-BCDataCache
Get-BCDataCacheExtension
Get-BCHashCache
Get-BCHostedCacheServer-
Configuration
Get-BCNetworkConfigura-
tion
Get-BCStatus
Get-BitLockerVolume
Get-ClusteredScheduled-
Task
Get-CounterSample
Get-DAClientExperience-
Configuration
Get-DAConnectionStatus
Get-DAEntryPointTableItem
Get-Disk
Get-DiskImage
Get-DisplayResolution
Get-DnsClient
Get-DnsClientCache
Get-DnsClientGlobalSet-
ting
Get-DnsClientNrptGlobal
Get-DnsClientNrptPolicy
Get-DnsClientNrptRule
Get-DnsClientServerAd-
dress
Get-Dtc
Get-DtcAdvancedHostSet-
ting
Get-DtcAdvancedSetting
Get-DtcClusterDefault
Get-DtcClusterTMMapping
Get-DtcDefault
Get-DtcLog
Get-DtcNetworkSetting
Get-DtcTransaction
Get-DtcTransactionsSta-
tistics
Get-DtcTransactionsTrace-
Session
Get-DtcTransactionsTrace-
Setting
Get-FileIntegrity
Get-InitiatorId
Get-InitiatorPort
Get-IscsiConnection
Get-IscsiSession
Get-IscsiTarget
Get-IscsiTargetPortal
Get-IseSnippet
Get-LogProperties
Get-MaskingSet
Get-MMAgent
Get-NCSIPolicyConfigura-
tion
Get-Net6to4Configuration
Get-NetAdapter
Get-NetAdapterAdvanced-
Property
Get-NetAdapterBinding
Get-NetAdapterChecksumOf-
fload
Get-NetAdapterEncapsula-
tedPacketTaskOffload
Get-NetAdapterHardwareIn-
fo
Get-NetAdapterIPsecOf-
fload
Get-NetAdapterLso
Get-NetAdapterPowerMana-
gement
Get-NetAdapterQos
Get-NetAdapterRdma
Get-NetAdapterRsc
Get-NetAdapterRss
Get-NetAdapterSriov
Get-NetAdapterSriovVf
Get-NetAdapterStatistics
Get-NetAdapterVmq
Get-NetAdapterVmqQueue
Get-NetAdapterVPort
Get-NetConnectionProfile
Get-NetDnsTransitionCon-
figuration
Get-NetDnsTransitionMo-
nitoring
Get-NetFirewallAddress-
Filter
Get-NetFirewallApplica-
tionFilter
Get-NetFirewallInterface-
Filter
Get-NetFirewallInterface-
TypeFilter
Get-NetFirewallPortFilter
Get-NetFirewallProfile
Get-NetFirewallRule
Get-NetFirewallSecurity-
Filter
Get-NetFirewallService-
Filter
Get-NetFirewallSetting
Get-NetIPAddress
Get-NetIPConfiguration
Get-NetIPHttpsConfigura-
tion
```

```
Get-NetIPHttpsState
Get-NetIPInterface
Get-NetIPsecDospSetting
Get-NetIPsecMainModeCryp-
toSet
Get-NetIPsecMainModeRule
Get-NetIPsecMainModeSA
Get-NetIPsecPhase1AuthSet
Get-NetIPsecPhase2AuthSet
Get-NetIPsecQuickMode-
CryptoSet
Get-NetIPsecQuickModeSA
Get-NetIPsecRule
Get-NetIPv4Protocol
Get-NetIPv6Protocol
Get-NetIsatapConfigura-
tion
Get-NetLbfoTeam
Get-NetLbfoTeamMember
Get-NetLbfoTeamNic
Get-NetNatTransitionCon-
figuration
Get-NetNatTransitionMo-
nitoring
Get-NetNeighbor
Get-NetOffloadGlobalSet-
ting
Get-NetPrefixPolicy
Get-NetQosPolicy
Get-NetRoute
Get-NetSwitchTeam
Get-NetSwitchTeamMember
Get-NetTCPConnection
Get-NetTCPSetting
Get-NetTeredoConfigura-
tion
Get-NetTeredoState
Get-NetTransportFilter
Get-NetUDPEndpoint
Get-NetUDPSetting
Get-NfsClientConfigura-
tion
Get-NfsClientgroup
Get-NfsClientLock
Get-NfsMappingStore
Get-NfsMountedClient
Get-NfsNetgroupStore
Get-NfsOpenFile
Get-NfsServerConfigura-
tion
Get-NfsSession
Get-NfsShare
Get-NfsSharePermission
Get-NfsStatistics
Get-OdbcDriver
Get-OdbcDsn
Get-OdbcPerfCounter
Get-OffloadDataTransfer-
Setting
Get-Partition
Get-PartitionSupported-
Size
Get-PerformanceCollector
Get-PhysicalDisk
Get-PrintConfiguration
Get-Printer
Get-PrinterDriver
Get-PrinterPort
Get-PrinterProperty
Get-PrintJob
Get-RDAvailableApp
Get-RDCertificate
Get-RDConnectionBrokerHi-
ghAvailability
Get-RDDeploymentGateway-
Configuration
Get-RDFileTypeAssociation
Get-RDLicenseConfigura-
tion
Get-RDPersonalVirtual-
DesktopAssignment
Get-RDPersonalVirtual-
DesktopPatchSchedule
Get-RDRemoteApp
Get-RDRemoteDesktop
Get-RDServer
Get-RDSessionCollection
Get-RDSessionCollection-
Configuration
Get-RDSessionHost
Get-RDUserSession
Get-RDVirtualDesktop
Get-RDVirtualDesktopCol-
lection
Get-RDVirtualDesktopCol-
lectionConfiguration
Get-RDVirtualDesktopCol-
lectionJobStatus
Get-RDVirtualDesktopCon-
currency
Get-RDVirtualDesktopIdle-
Count
Get-RDVirtualDesktopTem-
plateExportPath
Get-RDWorkspace
Get-ResiliencySetting
Get-ScheduledTask
Get-ScheduledTaskInfo
Get-ServerBpaResult
Get-ServerClusterName
Get-ServerEvent
Get-ServerFeature
Get-ServerInventory
Get-ServerService
Get-SmbClientConfigura-
tion
Get-SmbClientNetworkIn-
terface
Get-SmbConnection
Get-SmbMapping
Get-SmbMultichannelCon-
nection
Get-SmbMultichannelCons-
traint
Get-SmbOpenFile
Get-SmbServerConfigura-
tion
Get-SmbServerNetworkIn-
terface
Get-SmbSession
Get-SmbShare
Get-SmbShareAccess
Get-SmbWitnessClient
Get-StorageJob
Get-StoragePool
Get-StorageProvider
Get-StorageReliability-
Counter
Get-StorageSetting
Get-StorageSubSystem
Get-SupportedClusterSizes
Get-SupportedFileSystems
Get-TargetPort
Get-TargetPortal
Get-Ual
Get-UalDailyAccess
Get-UalDailyDeviceAccess
Get-UalDailyUserAccess
Get-UalDeviceAccess
Get-UalDns
Get-UalHyperV
Get-UalOverview
Get-UalServerDevice
Get-UalServerUser
Get-UalSystemId
Get-UalUserAccess
Get-Verb
Get-VirtualDisk
Get-VirtualDiskSupported-
Size
Get-Volume
Get-VolumeCorruptionCount
Get-VolumeScrubPolicy
Get-VpnConnection
Get-WdacBidTrace
Get-WindowsDeveloperLi-
cense
Get-WindowsFeature
Grant-NfsSharePermission
Grant-RDOUAccess
Grant-SmbShareAccess
H:
```

help
Hide-VirtualDisk
I:
Import-BCCachePackage
Import-BCSecretKey
Import-IseSnippet
Import-RDPersonalVirtual-DesktopAssignment
ImportSystemModules
Initialize-Disk
Install-Dtc
Install-WindowsFeature
Invoke-AsWorkflow
Invoke-RDUserLogoff
J:
K:
L:
Lock-BitLocker
M:
mkdir
more
Mount-DiskImage
Move-RDVirtualDesktop
Move-SmbWitnessClient
N:
New-DAEntryPointTableItem
New-EapConfiguration
New-IscsiTargetPortal
New-IseSnippet
New-MaskingSet
New-NetAdapterAdvanced-Property
New-NetFirewallRule
New-NetIPAddress
New-NetIPHttpsConfigura-tion
New-NetIPsecDospSetting
New-NetIPsecMainModeCryp-toSet
New-NetIPsecMainModeRule
New-NetIPsecPhase1AuthSet
New-NetIPsecPhase2AuthSet
New-NetIPsecQuickMode-CryptoSet
New-NetIPsecRule
New-NetLbfoTeam
New-NetNatTransitionCon-figuration
New-NetNeighbor
New-NetQosPolicy
New-NetRoute
New-NetSwitchTeam
New-NetTransportFilter
New-NfsClientgroup
New-NfsShare
New-Partition
New-PSWorkflowSession
New-RDCertificate
New-RDPersonalVirtual-DesktopPatchSchedule
New-RDRemoteApp
New-RDSessionCollection
New-RDSessionDeployment
New-RDVirtualDesktopCol-lection
New-RDVirtualDesktopDe-ployment
New-ScheduledTask
New-ScheduledTaskAction
New-ScheduledTaskPrinci-pal
New-ScheduledTaskSetting-sSet
New-ScheduledTaskTrigger
New-SmbMapping
New-SmbMultichannelCons-traint
New-SmbShare
New-StoragePool
New-StorageSubsystemVir-tualDisk
New-VirtualDisk
New-VirtualDiskClone
New-VirtualDiskSnapshot
O:
Open-NetGPO
Optimize-Volume
oss
P:
Pause
prompt
Publish-BCFileContent
Publish-BCWebContent
Q:
R:
Register-ClusteredSchedu-ledTask
Register-DnsClient
Register-IscsiSession
Register-ScheduledTask
Remove-BCDataCacheExten-sion
Remove-BitLockerKeyPro-tector
Remove-DAEntryPointTa-bleItem
Remove-DnsClientNrptRule
Remove-DtcClusterTMMap-ping
Remove-InitiatorId
Remove-InitiatorIdFrom-MaskingSet
Remove-IscsiTargetPortal
Remove-MaskingSet
Remove-NetAdapterAdvan-cedProperty
Remove-NetFirewallRule
Remove-NetIPAddress
Remove-NetIPHttpsCertBin-ding
Remove-NetIPHttpsConfigu-ration
Remove-NetIPsecDospSet-ting
Remove-NetIPsecMainMode-CryptoSet
Remove-NetIPsecMainMode-Rule
Remove-NetIPsecMainModeSA
Remove-NetIPsec-Phase1AuthSet
Remove-NetIPsec-Phase2AuthSet
Remove-NetIPsecQuickMode-CryptoSet
Remove-NetIPsecQuickMode-SA
Remove-NetIPsecRule
Remove-NetLbfoTeam
Remove-NetLbfoTeamMember
Remove-NetLbfoTeamNic
Remove-NetNatTransition-Configuration
Remove-NetNeighbor
Remove-NetQosPolicy
Remove-NetRoute
Remove-NetSwitchTeam
Remove-NetSwitchTeamMem-ber
Remove-NetTransportFilter
Remove-NfsClientgroup
Remove-NfsShare
Remove-OdbcDsn
Remove-Partition
Remove-PartitionAccess-Path
Remove-PhysicalDisk
Remove-Printer
Remove-PrinterDriver
Remove-PrinterPort
Remove-PrintJob
Remove-RDPersonalVirtual-DesktopAssignment
Remove-RDPersonalVirtual-DesktopPatchSchedule
Remove-RDRemoteApp
Remove-RDServer
Remove-RDSessionCollec-tion
Remove-RDSessionHost
Remove-RDVirtualDesktop-Collection
Remove-RDVirtualDesktop-FromCollection

```
Remove-ServerPerformance-
Log
Remove-SmbMapping
Remove-SmbMultichannel-
Constraint
Remove-SmbShare
Remove-StoragePool
Remove-TargetPortFromMas-
kingSet
Remove-VirtualDisk
Remove-VirtualDiskFrom-
MaskingSet
Remove-VpnConnection
Rename-DAEntryPointTa-
bleItem
Rename-MaskingSet
Rename-NetAdapter
Rename-NetFirewallRule
Rename-NetIPHttpsConfigu-
ration
Rename-NetIPsecMainMode-
CryptoSet
Rename-NetIPsecMainMode-
Rule
Rename-NetIPsec-
Phase1AuthSet
Rename-NetIPsec-
Phase2AuthSet
Rename-NetIPsecQuickMode-
CryptoSet
Rename-NetIPsecRule
Rename-NetLbfoTeam
Rename-NetSwitchTeam
Rename-NfsClientgroup
Rename-Printer
Repair-FileIntegrity
Repair-VirtualDisk
Repair-Volume
Reset-BC
Reset-DAClientExperience-
Configuration
Reset-DAEntryPointTableI-
tem
Reset-DtcLog
Reset-NCSIPolicyConfigu-
ration
Reset-Net6to4Configura-
tion
Reset-NetAdapterAdvanced-
Property
Reset-NetDnsTransition-
Configuration
Reset-NetIPHttpsConfigu-
ration
Reset-NetIsatapConfigura-
tion
Reset-NetTeredoConfigura-
tion
Reset-NfsStatistics
Reset-PhysicalDisk
Reset-StorageReliability-
Counter
Resize-Partition
Resize-VirtualDisk
Resolve-NfsMappedIdentity
Restart-NetAdapter
Restart-PrintJob
Resume-BitLocker
Resume-PrintJob
Revoke-NfsClientLock
Revoke-NfsMountedClient
Revoke-NfsOpenFile
Revoke-NfsSharePermission
Revoke-SmbShareAccess
S:
Save-NetGPO
Send-RDUserMessage
Set-BCAuthentication
Set-BCCache
Set-BCDataCacheEntry-
MaxAge
Set-BCMinSMBLatency
Set-BCSecretKey
Set-ClusteredScheduled-
Task
Set-DAClientExperience-
Configuration
Set-DAEntryPointTableItem
Set-Disk
Set-DisplayResolution
Set-DnsClient
Set-DnsClientGlobalSet-
ting
Set-DnsClientNrptGlobal
Set-DnsClientNrptRule
Set-DnsClientServerAd-
dress
Set-DtcAdvancedHostSet-
ting
Set-DtcAdvancedSetting
Set-DtcClusterDefault
Set-DtcClusterTMMapping
Set-DtcDefault
Set-DtcLog
Set-DtcNetworkSetting
Set-DtcTransaction
Set-DtcTransactionsTrace-
Session
Set-DtcTransactionsTrace-
Setting
Set-FileIntegrity
Set-InitiatorPort
Set-IscsiChapSecret
Set-LogProperties
Set-MMAgent
Set-NCSIPolicyConfigura-
tion
Set-Net6to4Configuration
Set-NetAdapter
Set-NetAdapterAdvanced-
Property
Set-NetAdapterBinding
Set-NetAdapterChecksumOf-
fload
Set-NetAdapterEncapsula-
tedPacketTaskOffload
Set-NetAdapterIPsecOf-
fload
Set-NetAdapterLso
Set-NetAdapterPowerMana-
gement
Set-NetAdapterQos
Set-NetAdapterRdma
Set-NetAdapterRsc
Set-NetAdapterRss
Set-NetAdapterSriov
Set-NetAdapterVmq
Set-NetConnectionProfile
Set-NetDnsTransitionCon-
figuration
Set-NetFirewallAddress-
Filter
Set-NetFirewallApplica-
tionFilter
Set-NetFirewallInterface-
Filter
Set-NetFirewallInterface-
TypeFilter
Set-NetFirewallPortFilter
Set-NetFirewallProfile
Set-NetFirewallRule
Set-NetFirewallSecurity-
Filter
Set-NetFirewallService-
Filter
Set-NetFirewallSetting
Set-NetIPAddress
Set-NetIPHttpsConfigura-
tion
Set-NetIPInterface
Set-NetIPsecDospSetting
Set-NetIPsecMainModeCryp-
toSet
Set-NetIPsecMainModeRule
Set-NetIPsecPhase1AuthSet
Set-NetIPsecPhase2AuthSet
Set-NetIPsecQuickMode-
CryptoSet
Set-NetIPsecRule
Set-NetIPv4Protocol
Set-NetIPv6Protocol
Set-NetIsatapConfigura-
tion
```

```
Set-NetLbfoTeam
Set-NetLbfoTeamMember
Set-NetLbfoTeamNic
Set-NetNatTransitionCon-
figuration
Set-NetNeighbor
Set-NetOffloadGlobalSet-
ting
Set-NetQosPolicy
Set-NetRoute
Set-NetTCPSetting
Set-NetTeredoConfigura-
tion
Set-NetUDPSetting
Set-NfsClientConfigura-
tion
Set-NfsClientgroup
Set-NfsMappingStore
Set-NfsNetgroupStore
Set-NfsServerConfigura-
tion
Set-NfsShare
Set-OdbcDriver
Set-OdbcDsn
Set-Partition
Set-PhysicalDisk
Set-PrintConfiguration
Set-Printer
Set-PrinterProperty
Set-RDActiveManagement-
Server
Set-RDCertificate
Set-RDClientAccessName
Set-RDConnectionBrokerHi-
ghAvailability
Set-RDDatabaseConnectionS-
tring
Set-RDDeploymentGateway-
Configuration
Set-RDFileTypeAssociation
Set-RDLicenseConfigura-
tion
Set-RDPersonalVirtual-
DesktopAssignment
Set-RDPersonalVirtual-
DesktopPatchSchedule
Set-RDRemoteApp
Set-RDRemoteDesktop
Set-RDSessionCollection-
Configuration
Set-RDSessionHost
Set-RDVirtualDesktopCol-
lectionConfiguration
Set-RDVirtualDesktopCon-
currency
Set-RDVirtualDesktopIdle-
Count
Set-RDVirtualDesktopTem-
plateExportPath
Set-RDWorkspace
Set-ResiliencySetting
Set-ScheduledTask
Set-SmbClientConfigura-
tion
Set-SmbServerConfigura-
tion
Set-SmbShare
Set-StoragePool
Set-StorageSetting
Set-StorageSubSystem
Set-VirtualDisk
Set-Volume
Set-VolumeScrubPolicy
Set-VpnConnection
Set-VpnConnectionProxy
Show-NetFirewallRule
Show-NetIPsecRule
Show-VirtualDisk
Show-WindowsDeveloperLi-
censeRegistration
Start-Dtc
Start-DtcTransactionsTra-
ceSession
Start-PerformanceCollec-
tor
Start-ScheduledTask
Start-Trace
Stop-Dtc
Stop-DtcTransactionsTra-
ceSession
Stop-PerformanceCollector
Stop-RDVirtualDesktopCol-
lectionJob
Stop-ScheduledTask
Stop-Trace
Suspend-BitLocker
Suspend-PrintJob
Sync-NetIPsecRule
T:
TabExpansion2
Test-Dtc
Test-NfsMappingStore
Test-RDOUAccess
Test-RDVirtualDesktopAD-
MachineAccountReuse
U:
Unblock-SmbShareAccess
Uninstall-Dtc
Uninstall-WindowsFeature
Unlock-BitLocker
Unregister-ClusteredSche-
duledTask
Unregister-IscsiSession
Unregister-ScheduledTask
Unregister-WindowsDevelo-
perLicense
Update-Disk
Update-HostStorageCache
Update-IscsiTarget
Update-IscsiTargetPortal
Update-NetIPsecRule
Update-RDVirtualDesktop-
Collection
Update-SmbMultichannel-
Connection
Update-StorageProvider-
Cache
V:
W:
Write-DtcTransactionsTra-
ceSession
X:
Y:
Z:
```

XVI.Annexe 3 : de Vbs à Powershell, documentation adaptée d'un document Microsoft

VBScript Function	Windows PowerShell Equivalent
Abs	$a = [math]::abs(-15)
Array	$a = "red","orange","yellow","green","blue","indigo","violet"
Asc	$a = [byte][char] "A"
Atn	$a = [math]::atan(90)
CBool	$a = 0 $a = [bool] $a
CByte	$a = "11.45" $a = [byte] $a
CCur	$a = "{0:C}" -f 13
CDate	$a = "11/1/2006" $a = [datetime] $a
CDbl	$a = "11.45" $a = [double] $a
Chr	$a = [char]34
CInt	$a = "11.57" $a = [int] $a
CLng	$a = "123456789.45" $a = [long] $a
Cos	$a = [math]::cos(45)
CreateObject	$a.visible = $True $a = new-object -comobject Excel.Application -strict
CSng	$a = "11.45" $a = [single] $a
CStr	$a = 17 $a = [string] $a
Date	$a = get-date –format d
DateAdd	$a = (get-date).AddDays(37) (get-date).AddHours(37) (get-date).AddMilliseconds(37) (get-date).AddMinutes(37) (get-date).AddMonths(37) (get-date).AddSeconds(37) (get-date).AddTicks(37) (get-date).AddYears(37) $a = ((get-date).AddHours(2)).AddMinutes(34)
DateDiff	$a = New-TimeSpan $(Get-Date) $(Get-Date –month 12 -day 31 -year 2006 -hour 23 -minute 30) $a.Days Days : 109 Hours : 3 Minutes : 55 Seconds : 0 Milliseconds : 0 Ticks : 94317000000000 TotalDays : 109.163194444444 TotalHours : 2619.91666666667 TotalMinutes : 157195 TotalSeconds : 9431700 TotalMilliseconds : 9431700000

DatePart	$a = (get-date).day $a = (get-date).dayofweek $a = (get-date).dayofyear $a = (get-date).hour $a = (get-date).millisecond $a = (get-date).minute $a = (get-date).month $a = (get-date).second $a = (get-date).timeofday $a = (get-date).year $a = (get-date).hour
DateSerial	MyDate1 = DateSerial(2006, 12, 31) $a = get-date -y 2006 -mo 12 -day 31
DateValue	$a = [datetime] "12/1/2006"
Day	$a = (get-date).day
Eval	$a = 2 + 2 -eq 45
Exp	$a = [math]::exp(2)
Filter	$a = "Monday","Month","Merry","Mansion","Modest" $b = ($a | where-object {$_ -like "Mon*"})
FormatCurrency	$a = 1000 $a = "{0:C}" -f $a
FormatDateTime	$a = (get-date).tolongdatestring() $a = (get-date).toshortdatestring() $a = (get-date).tolongtimestring() $a = (get-date).toshorttimestring()
FormatNumber	$a = 11 $a = "{0:N6}" -f $a
FormatPercent	$a = .113 $a = "{0:P1}" -f $a
GetLocale	$a = (get-culture).lcid $a = (get-culture).displayname
Hex	$a = 4517 $a = "{0:X}" -f $a
Hour	$a = (get-date).hour
InputBox	$a = new-object -comobject MSScriptControl.ScriptControl $a.language = "vbscript" $a.addcode("function getInput() getInput = inputbox(`"Message box prompt`",`"Message Box Title`") end function") $b = $a.eval("getInput")
InStr	$a = "wombat" $b = $a.contains("m") $b = $a.indexof("m")
InStrRev	$a = "1234x6789x1234" $b = $a.lastindexofany("x")
Int/Fix	$a = 11.98 $a = [math]::truncate($a)
IsArray	$a = 22,5,10,8,12,9,80 $b = $a -is [array]
IsDate	$a = 11/2/2006 $a -is [datetime] $a = [datetime] "11/2/2006"
IsEmpty	$a = "" $b = $a.length -eq 0
IsNull	$a = $z -eq $null
IsNumeric	$a = 44.5

	[reflection.assembly]::LoadWithPartialName("'Microsoft.VisualBasic") $b = [Microsoft.VisualBasic.Information]::isnumeric($a)
IsObject	$a = new-object -comobject scripting.filesystemobject $b = $a -is [object]
Join	$a = "h","e","l","l","o" $b = [string]::join("", $a)
LBound	$a = 1,2,3,4,5,6,7,8,9 $b = $a.getlowerbound(0)
LCase	$a = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" $a = $a.ToLower()
Left	$a="ABCDEFGHIJKLMNOPQRSTUVWXYZ" $a = $a.substring(0,3)
Len	$a = "abcdefghijklmnopqrstuvwxyz" $b = $a.length
Log	$a = [math]::log(100)
LTrim	$a = "..........123456789.........." $a = $a.TrimStart()
RTrim	$a = "..........123456789.........." $a = $a.TrimEnd()
Trim	$a = "..........123456789.........." $a = $a.Trim()
Mid	$a="ABCDEFG" $a = $a.substring(2,3)
Minute	$a =(get-date).minute
Month	$a = get-date -f "MM" $a = [int] (get-date -f "MM")
MonthName	$a = get-date -f "MMMM"
MsgBox	$a = new-object -comobject wscript.shell $b = $a.popup("This is a test",0,"Test Message Box",1)
Now	$a = get-date
Oct	$a = [System.Convert]::ToString(999,8)
Replace	$a = "bxnxnx" $a = $a -replace("x","a")
RGB	$blue = 10 $green= 10 $red = 10 $a = [long] ($blue + ($green * 256) + ($red * 65536))
Right	$a = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" $a = $a.substring($a.length - 9, 9)
Rnd	$a = new-object random $b = $a.next(1,100) $b = $a.next()
Round	$a = [math]::round(45.987654321, 2)
ScriptEngine	$a = (get-host).version
ScriptEngineBuildVersion	$a = (get-host).version.build
ScriptEngineMajorVersion	$a = (get-host).version.major
ScriptEngineMinorVersion	$a = (get-host).version.minor
Second	$a = (get-date).second
Sgn	$a = [math]::sign(-453)
Sin	$a = [math]::sin(45)
Space	$a = " " * 25 $a = $a + "x"
Split	$a = "atl-ws-01,atl-ws-02,atl-ws-03,atl-ws-04"

	$b = $a.split(",")
Sqr	$a = [math]::sqrt(144)
StrComp	$a = "dog" $b = "DOG" $c = [String]::Compare($a,$b,$True)
String	$a = "=" * 20
StrReverse	$a = "Scripting Guys" for ($i = $a.length - 1; $i -ge 0; $i--) {$b = $b + ($a.substring($i,1))}
Tan	$a = [math]::tan(45)
Time	$a = get-date -displayhint time
TimeSerial	$a = get-date -h 17 -mi 10 -s 45 -displayhint time
TimeValue	$a = [datetime] "1:45 AM"
TypeName	**$a = 55.86768** **$b = $a.GetType().name**
UBound	$a = "a","b","c","d","e" $a.getupperbound(0) $a.length-1
UCase	$a = "abcdefghijklmnopqrstuvwxyz" $a = $a.ToUpper()
WeekdayName	$a = (get-date).dayofweek $a = (get-date "12/25/2007").dayofweek
Year	$a = (get-date).year $a = (get-date "9/15/2005").year
Const Statement	set-variable -name ForReading -value 1 -option constant
Dim Statement	$a = [string]
Execute Statement	$a = "get-date" invoke-expression $a
Function Statement	function multiplynumbers { $args[0] * $args[1] } multiplynumbers 38 99
On Error Statement	$erroractionpreference = "SilentlyContinue" Incidentally, your choices for this variable include: SilentlyContinue Continue (the default value) Inquire Stop
Option Explicit Statement	set-psdebug –strict set-psdebug -off
Private Statement	$Private:a = 5
Public Statement	$Global:a = 199
Randomize Statement	$a = new-object random $b = $a.next()
ReDim Statement	$a = 1,2,3,4,5 $a = $a + 100 $a = $a[0..2]
Set Statement	$a = new-object -comobject Excel.Application $a.visible = $True
Stop Statement	set-psdebug –step set-psdebug -off
Sub Statement	function multiplynumbers { $args[0] * $args[1] } multiplynumbers 38 99
Description Property	$a = $error[0].ToString()
HelpContext Property	$a = $error[0].helplink
HelpFile Property	$a = $error[0].helplink
Number Property	ScriptHalted $error[0].errorrecord

Source Property	$a = $error[0].source
Clear Method	$error[0] = "" $error.clear()
Raise Method	$b = "The file could not be found."; throw $b

XVII.Annexe 4 : opérateurs Where-Object

EqualSet	EQ
ScriptBlockSet	ScriptBlock
CaseSensitiveGreaterThanSet	CGT
CaseSensitiveNotEqualSet	GNE
LessThanSet	LT
CaseSensitiveEqualSet	CEQ
NotEqualSet	NE
GreaterThanSet	GT
CaseSensitiveLessThanSet	CLT
GreaterOrEqualSet	GE
CaseSensitiveGreaterOrEqualSet	CGE
LessOrEqualSet	LE
CaseSensitiveLessOrEqualSet	CLE
LikeSet	Like
CaseSensitiveLikeSet	CLike
NotLikeSet	NotLike
CaseSensitiveNotLikeSet	CNotLike
MatchSet	Match
CaseSensitiveMatchSet	CMatch
NotMatchSet	NotMatch
CaseSensitiveNotMatchSet	CNotMatch
ContainsSet	Contains
CaseSensitiveContainsSet	CContains
NotContainsSet	NotContains
CaseSensitiveNotContainsSet	CNotContains
InSet	In
CaseSensitiveInSet	CIn
NotInSet	NotIn
CaseSensitiveNotInSet	CNotIn
IsSet	Is
IsNotSet	IsNot

