

Table des matières

I.	Présentation.....	6
A.	Préambule.....	6
B.	Technologies de scripting.....	6
C.	PowerShell 3.....	6
D.	Les outils.....	6
II.	Premiers pas.....	7
A.	Les applets de commande ou cmdlets.....	7
B.	L'interpréteur.....	7
C.	Protection.....	7
1.	Le niveau de sécurité : Get-ExecutionPolicy.....	7
2.	Changer le niveau de sécurité : Set-ExecutionPolicy.....	7
3.	Signature.....	7
4.	Voir aussi.....	7
5.	Stratégies.....	8
6.	Autorité de certification.....	8
7.	Associer un certificate à un script.....	8
D.	Aide.....	8
1.	Informations de plate-forme : Get-Host.....	8
2.	La liste des commandes : Get-Command.....	8
3.	L'aide : Get-Help.....	8
4.	Actualiser l'aide 3.0.....	8
5.	Méthodes et propriétés associées à une cmdlet.....	8
6.	Afficher les propriétés d'un cmdlet.....	9
7.	Mode GUI.....	9
8.	Afficher les méthodes et propriétés d'un objet.....	9
9.	Les fournisseurs PowerShell : Get-PSPProvider.....	9
10.	Historique.....	9
E.	Exécution des scripts.....	9
1.	Exécution d'un script.....	9
2.	Appel d'un autre script.....	9
3.	Récupération du contenu de l'exécution d'une commande système.....	9
4.	Mac Address.....	9
5.	Variable d'environnement.....	10
6.	Appel d'un programme.....	10
7.	Mesurer le temps d'exécution : Measure-Command.....	10
8.	Tempo.....	10
9.	Trigger.....	10
10.	Envoi de mail.....	10
F.	Historique.....	11
1.	Visualiser l'historique.....	11
2.	Récupérer l'historique.....	11
3.	Exécuter une commande de l'historique.....	11
4.	Voir aussi.....	11
G.	Informations de langue.....	11
H.	Passage d'arguments.....	11
1.	Par tableau.....	11
2.	Par la méthode Param.....	11
I.	Commentaires.....	11
J.	Instruction sur plusieurs lignes.....	11

III.	Cmdlets système.....	12
A.	Le journal d'événements.....	12
B.	Les services.....	12
1.	La liste des services.....	12
2.	Démarrer, arrêter un service.....	12
3.	Mettre en suspens, reprendre un service.....	12
4.	Modifier les propriétés des services.....	12
C.	Les process.....	12
1.	Liste des process.....	12
2.	Arrêter un process.....	12
3.	Verbosité/Erreur.....	12
D.	Informations.....	13
E.	CIM.....	13
F.	WMI.....	13
IV.	Eléments du langage.....	14
A.	Les variables et les constantes.....	14
1.	Les variables.....	14
2.	Les types.....	14
3.	Les chaînes.....	14
4.	Caractères spéciaux.....	14
5.	Substitution de variables.....	14
6.	Les variables prédéfinies.....	14
7.	Les constantes.....	15
B.	Les tableaux.....	15
1.	Principes de base.....	15
2.	Exemple.....	15
3.	Effacer un élément avec méthode .Net.....	16
4.	Tableaux associatifs.....	16
5.	Autres méthodes.....	16
6.	Portée.....	16
C.	Nombre aléatoire.....	16
D.	Opérateurs.....	16
1.	Concaténation.....	16
2.	Comparaison.....	16
3.	Expressions régulières.....	16
4.	Logiques.....	17
5.	Plages.....	17
6.	Appartenance.....	17
7.	Opérateurs binaires.....	17
8.	Affectation.....	17
9.	Cast.....	17
10.	Forcer la définition de variables.....	17
E.	Structures de contrôle.....	17
1.	Do.....	17
2.	While.....	17
3.	For.....	18
4.	Break.....	18
5.	If.....	18
6.	Foreach.....	18
7.	Switch.....	18
8.	Exemple conditionnelle.....	19
F.	Gestion d'erreurs.....	19

1.	Préférence.....	19
2.	Cas par cas.....	19
3.	Trap.....	19
4.	Try...Catch.....	19
5.	Débogage.....	19
G.	Pipelining.....	20
1.	Comptage.....	20
2.	Stats.....	20
3.	Sélection.....	20
4.	Tri.....	20
5.	Différence.....	20
6.	Affichage.....	20
7.	Filtre.....	21
8.	Valeurs unique.....	21
9.	Propriétés.....	21
10.	Impressions.....	21
11.	Boucle.....	21
12.	Tri.....	21
13.	Message.....	22
14.	Interaction.....	22
H.	Fonctions.....	22
1.	Sans retour.....	22
2.	Avec retour.....	22
I.	Gestion des modules.....	22
1.	Emplacement des modules.....	22
2.	Télécharger des modules complémentaires.....	23
3.	Les modules liés à l'administration.....	23
4.	Commandes d'un module.....	23
5.	Charger automatiquement les modules.....	23
6.	Décharger un module.....	23
7.	Créer un module.....	23
8.	Exemple : devices.psm1.....	23
V.	Gestion des heures et des dates.....	24
A.	Obtenir la date et l'heure : Get-Date.....	24
B.	Méthodes associées à la cmdlet Get-Date.....	24
C.	Changer la date et l'heure : Set-Date.....	24
D.	Calculs sur date.....	24
E.	Création de fichier.....	24
VI.	Gestion des fichiers.....	25
A.	Système.....	25
1.	Copie de fichiers : Copy-Item.....	25
2.	Création de fichiers : New-Item.....	25
3.	Déplacer les fichiers.....	25
4.	Renommer les fichiers.....	25
1.	Suppression de fichiers : Remove-Item.....	25
B.	Informations sur les fichiers, répertoires et clés de registres.....	25
C.	Tester l'existence d'un chemin.....	25
D.	Lire un répertoire.....	25
1.	Commandes.....	25
2.	Attributs (IO.FileAttributes).....	26
E.	La sécurité.....	26

F.	Ajout à un fichier.....	26
G.	Recherche dans un fichier.....	26
H.	Les redirections.....	26
I.	Création d'un fichier.....	26
J.	Effacer le contenu d'un fichier.....	26
K.	Convertir en Html.....	26
1.	Utiliser une page CSS.....	27
L.	Conversion en JSON.....	27
M.	Compter les lignes d'un fichier.....	27
N.	Lire un fichier CSV.....	27
O.	Les fichiers XML.....	27
P.	Export CSV.....	27
Q.	Sauvegarde d'un fichier.....	27
R.	Export Xml.....	27
S.	Sauvegarder dans un fichier texte.....	28
T.	Interactif.....	28
U.	Export / Import CSV Tableaux et Tableaux associatifs.....	28
VII.	Registre.....	29
A.	Lecture d'une clé.....	29
B.	Créer une clé.....	29
C.	Créer une valeur.....	29
D.	Suppression de clé.....	29
E.	Lecture / Ecriture.....	29
VIII.	Exécution distante.....	30
A.	Présentation.....	30
1.	Sécurité.....	30
B.	Authentification.....	30
C.	Machines de confiance (Poste à poste).....	30
D.	Droits.....	31
E.	Sessions.....	31
1.	Session temporaire.....	31
2.	Session permanente.....	31
3.	Exécution distante.....	31
4.	Rappel de la session.....	31
F.	Liste des commandes possibles.....	31
G.	Exemples.....	31
1.	Invoke-Command.....	31
2.	Get-Process.....	31
IX.	Modules Windows 8 et Windows 2012.....	32
A.	NetAdapter.....	32
1.	Importer le module NetAdapter.....	32
2.	Profil.....	32
3.	Lister les périphériques réseaux.....	32
4.	Elements attachés à la carte réseau.....	32
5.	Désactiver IPv6.....	32
B.	Partage réseau SmbShare.....	32
C.	Impression.....	32
D.	ODBC.....	32
E.	DNS.....	32
F.	Disque.....	32

G.	Drivers.....	33
H.	Applications.....	33
X.	A tester.....	34
A.	Panneau de configuration.....	34
B.	Renommer un ordinateur.....	34
XI.	Active Directory.....	35
A.	ADSI.....	35
1.	Gestion des groupes locaux.....	35
2.	Gestion des utilisateurs.....	35
B.	Module (à partir de Windows Server 2008).....	36
1.	Import.....	36
2.	Liste des lecteurs.....	36
3.	Gestion de l'annuaire.....	36
4.	Les utilisateurs.....	37
5.	Les groupes.....	37
C.	Déploiement (2012).....	38
1.	Ajout de la forêt.....	38
2.	Ajout du DC.....	38
3.	Désinstallation du DC.....	38
XII.	PowerShell sous Windows 2008 R2.....	39
A.	Source.....	39
B.	La listes des cmdlets.....	39
C.	La gestion des utilisateurs.....	42
D.	Les groupes.....	43
XIII.	Quelques exemples.....	44
A.	Liste des fichiers exécutés sur la machine.....	44
B.	Liste des services à partir du registre.....	44
C.	Utilisation des composants WSH Windows Scripting Host.....	44
1.	Wscript.Shell.....	44
2.	Wscript.Network.....	44
3.	Partage d'imprimante.....	45
4.	Scripting.FileSystemObject.....	45
D.	MySQL : lecture de tables.....	45
E.	Les compteurs.....	46
F.	MySQL : inventaire.....	46
1.	La table.....	46
2.	Le script.....	46
XIV.	Quelques sites.....	48
A.	Références.....	48
B.	Exemples de scripts.....	48
C.	Documentations.....	48
D.	Téléchargements.....	48
E.	Blogs.....	48
XV.	Annexe 1 : cmdlets et fonctions présentes sous Windows Server 2012.....	49
A.	Les CmdLets.....	49
B.	Les fonctions.....	52
XVI.	Annexe 3 : de Vbs à Powershell, documentation adaptée d'un document Microsoft.....	59

XVII. Annexe 4 : opérateurs Where-Object..... 64

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 avons 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 langages tels 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 fonctionnent, 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 certificat 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
$authentication = 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 $authentication
```

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.Verbose/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
gjjgj
'@
```

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	Événement 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	<code>\$MyInvocation.MyCommand.Name</code>

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 fonction, 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 Débogueur : 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 -AutoSize -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 "hkln:\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](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
$acIs=Get-Acl -Path 'c:\test\fictest.txt'
ForEach($fic in Get-ChildItem 'd:\powershell')
{
    $path=$fic.FullName
    Set-Acl -Path $path -AclObject $acIs
}
```

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

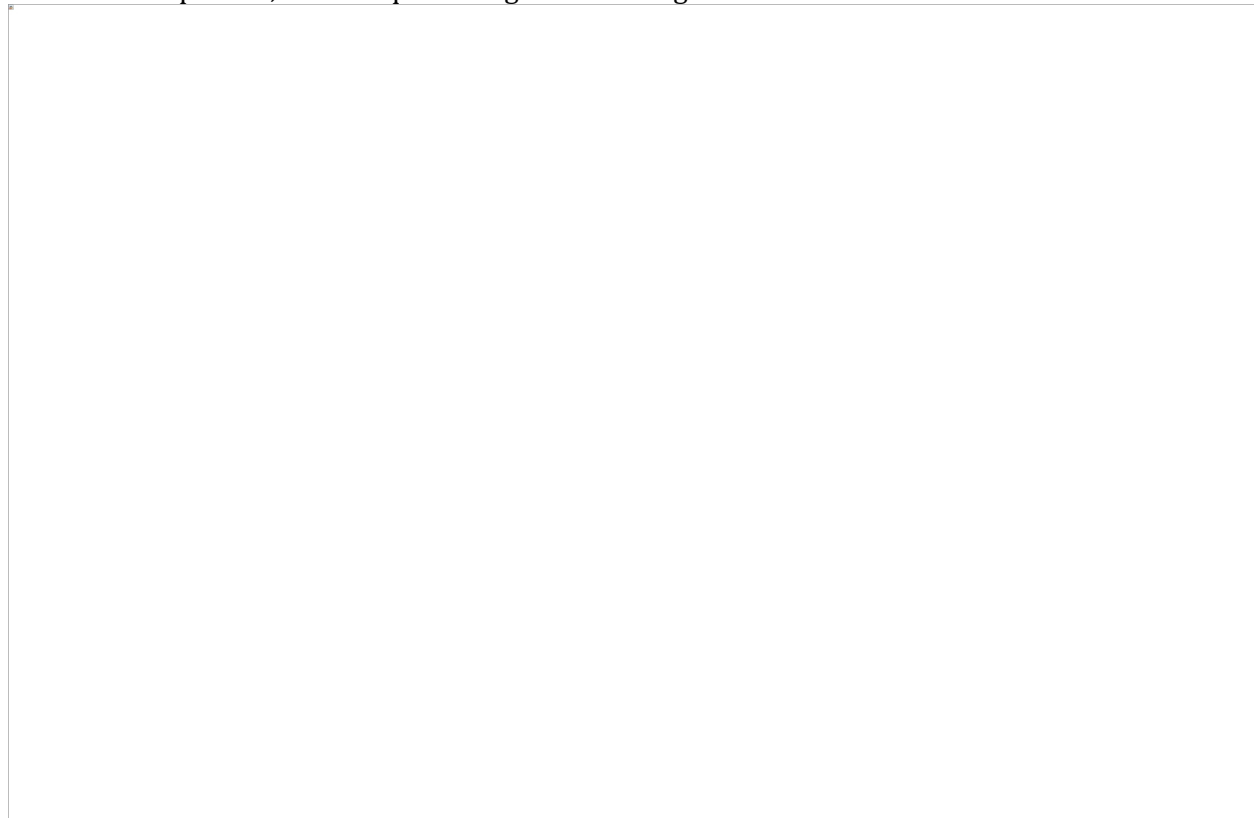
```
$val = Get-ItemProperty -Path  
hkml:software\microsoft\windows\currentversion\policies\system -Name "EnableLUA"  
if($val.EnableLUA -ne 0)  
{  
    set-itemproperty -Path hkml:software\microsoft\windows\currentversion\policies\system  
-Name "EnableLUA" -value 0  
}
```

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.Authentication

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 -Concatenate
```

```
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-ItemProperty -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
$authentication = New-Object System.Management.Automation.PSCredential `
("MF230\Administrateur",$motdepasse)
$session=New-PSSession -ComputerName MF230 -Credential $authentication
$path=Invoke-Command -ScriptBlock {$env:computername} -Session $session
$cmd=Invoke-Command -ScriptBlock {&ipconfig} -Session $session
clear
$path
$cmd
```

2.Get-Process

```
$motdepasse = ConvertTo-SecureString "password" -AsPlainText -Force
$authentication = New-Object System.Management.Automation.PSCredential `
("MF230\Administrateur",$motdepasse)
Enter-PSSession -ComputerName MF230 -Credential $authentication
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.Elementes 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://."
$grp= $conn.Create('group','test')
$grp.Put('Description','Groupe de test')
$grp.SetInfo()
$grp.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=dutout,DC=net' -name displayName
```

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

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=du-tout,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 utiliser 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
<u>Add-ADComputerServiceAccount</u>	Adds one or more service accounts to an Active Directory computer.
<u>Add-ADDomainControllerPasswordReplicationPolicy</u>	Adds users, computers, and groups to the Allowed List or the Denied List of the read-only domain controller (RODC) Password Replication Policy (PRP).
<u>Add-ADFineGrainedPasswordPolicySubject</u>	Applies a fine-grained password policy to one more users and groups.
<u>Add-ADGroupMember</u>	Adds one or more members to an Active Directory group.
<u>Add-ADPrincipalGroupMembership</u>	Adds a member to one or more Active Directory groups.
<u>Clear-ADAccountExpiration</u>	Clears the expiration date for an Active Directory account.
<u>Disable-ADAccount</u>	Disables an Active Directory account.
<u>Disable-ADOptionalFeature</u>	Disables an Active Directory optional feature.
<u>Enable-ADAccount</u>	Enables an Active Directory account.
<u>Enable-ADOptionalFeature</u>	Enables an Active Directory optional feature.
<u>Get-ADAccountAuthorizationGroup</u>	Gets the Active Directory security groups that contain an account.
<u>Get-ADAccountResultantPasswordReplicationPolicy</u>	Gets the resultant password replication policy for an Active Directory account.
<u>Get-ADComputer</u>	Gets one or more Active Directory computers.
<u>Get-ADComputerServiceAccount</u>	Gets the service accounts that are hosted by an Active Directory computer.
<u>Get-ADDefaultDomainPasswordPolicy</u>	Gets the default password policy for an Active Directory domain.
<u>Get-ADDomain</u>	Gets an Active Directory domain.
<u>Get-ADDomainController</u>	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.
<u>Get-ADDomainControllerPasswordReplicationPolicy</u>	Gets the members of the Allowed List or the Denied List of the RODC PRP.
<u>Get-ADDomainControllerPasswordReplicationPolicyUsage</u>	Gets the resultant password policy of the specified ADAccount on the specified RODC.
<u>Get-ADFineGrainedPasswordPolicy</u>	Gets one or more Active Directory fine-grained password policies.
<u>Get-ADFineGrainedPasswordPolicySubject</u>	Gets the users and groups to which a fine-grained password policy is applied.
<u>Get-ADForest</u>	Gets an Active Directory forest.
<u>Get-ADGroup</u>	Gets one or more Active Directory groups.
<u>Get-ADGroupMember</u>	Gets the members of an Active Directory group.
<u>Get-ADObject</u>	Gets one or more Active Directory objects.
<u>Get-ADOptionalFeature</u>	Gets one or more Active Directory optional features.
<u>Get-ADOrganizationalUnit</u>	Gets one or more Active Directory OUs.
<u>Get-ADPrincipalGroupMembership</u>	Gets the Active Directory groups that have a specified user, computer, or group.
<u>Get-ADRootDSE</u>	Gets the root of a domain controller information tree.
<u>Get-ADServiceAccount</u>	Gets one or more Active Directory service accounts.
<u>Get-ADUser</u>	Gets one or more Active Directory users.
<u>Get-ADUserResultantPasswordPolicy</u>	Gets the resultant password policy for a user.
<u>Install-ADServiceAccount</u>	Installs an Active Directory service account on a computer.
<u>Move-ADDirectoryServer</u>	Moves a domain controller in AD DS to a new site.
<u>Move-ADDirectoryServerOperationMasterRole</u>	Moves operation master (also known as flexible single master operations or FSMO) roles to an Active Directory domain controller.
<u>Move-ADObject</u>	Moves an Active Directory object or a container of objects to a different container or domain.
<u>New-ADComputer</u>	Creates a new Active Directory computer.
<u>New-ADFineGrainedPasswordPolicy</u>	Creates a new Active Directory fine-grained password policy.

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

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

New-ADGroup -name *GroupName* -GroupScope *Global|Universal* -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 lire les 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
{
}
```


3.Partage d'imprimante

4.Scripting.FileSystemObject

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

Get-Culture	Get-WinCultureFromLanguageListOptOut	Measure-Object
Get-DAPolicyChange	Get-WinDefaultInputMethodOverride	Merge-Partition
Get-Date	Get-WindowsDriver	Mount-IscsiVirtualDiskSnapshot
Get-Event	Get-WindowsEdition	Mount-WindowsImage
Get-EventLog	Get-WindowsErrorReporting	Move-Item
Get-EventSubscriber	Get-WindowsImage	Move-ItemProperty
Get-ExecutionPolicy	Get-WindowsOptionalFeature	New-Alias
Get-FormatData	Get-WindowsPackage	New-AppLockerPolicy
Get-Help	Get-WinEvent	New-CertificateNotificationTask
Get-History	Get-WinHomeLocation	New-CimInstance
Get-Host	Get-WinLanguageBarOption	New-CimSession
Get-HotFix	Get-WinSystemLocale	New-CimSessionOption
Get-IscsiServerTarget	Get-WinUILanguageOverride	New-DtcDiagnosticTransaction
Get-IscsiTargetServerSetting	Get-WinUserLanguageList	New-Event
Get-IscsiVirtualDisk	Get-WmiObject	New-EventLog
Get-IscsiVirtualDiskSnapshot	Get-WSManCredSSP	New-IscsiServerTarget
Get-Item	Get-WSManInstance	New-IscsiVirtualDisk
Get-ItemProperty	Group-Object	New-Item
Get-Job	Import-Alias	New-ItemProperty
Get-JobTrigger	Import-Certificate	New-JobTrigger
Get-KdsConfiguration	Import-Clixml	New-Module
Get-KdsRootKey	Import-Counter	New-ModuleManifest
Get-Location	Import-Csv	New-NetIPsecAuthProposal
Get-Member	Import-IscsiVirtualDisk	New-NetIPsecMainModeCryptoProposal
Get-Module	Import-LocalizedData	New-NetIPsecQuickModeCryptoProposal
Get-NfsMappedIdentity	Import-Module	New-NfsMappedIdentity
Get-NfsNetgroup	Import-PfxCertificate	New-NfsNetgroup
Get-PfxCertificate	Import-PSSession	New-Object
Get-PfxData	Import-TpmOwnerAuth	New-PSDrive
Get-Process	Initialize-Tpm	New-PSSession
Get-PSBreakpoint	Install-NfsMappingStore	New-PSSessionConfigurationFile
Get-PSCallStack	Invoke-ASCmd	New-PSSessionOption
Get-PSDrive	Invoke-BpaModel	New-PSTransportOption
Get-PSPProvider	Invoke-CimMethod	New-PSWorkflowExecutionOption
Get-PSSession	Invoke-Command	New-RestoreFolder
Get-PSSessionConfiguration	Invoke-Expression	New-RestoreLocation
Get-PSSnapin	Invoke-History	New-ScheduledJobOption
Get-Random	Invoke-Item	New-SelfSignedCertificate
Get-ScheduledJob	Invoke-PolicyEvaluation	New-Service
Get-ScheduledJobOption	Invoke-ProcessCube	New-SqlAvailabilityGroup
Get-SecureBootPolicy	Invoke-ProcessDimension	New-SqlAvailabilityGroupListener
Get-SecureBootUEFI	Invoke-ProcessPartition	New-SqlAvailabilityReplica
Get-Service	Invoke-RestMethod	New-SqlHADREndpoint
Get-Tpm	Invoke-Sqlcmd	New-TimeSpan
Get-TraceSource	Invoke-TroubleshootingPack	New-Variable
Get-Transaction	Invoke-WebRequest	New-WebServiceProxy
Get-TroubleshootingPack	Invoke-WmiMethod	New-WinEvent
Get-TypeData	Invoke-WSManAction	New-WinUserLanguageList
Get-UICulture	Join-DtcDiagnosticResourceManager	New-WSManInstance
Get-Unique	Join-Path	
Get-Variable	Join-SqlAvailabilityGroup	
Get-WheaMemoryPolicy	Limit-EventLog	
Get-WinAcceptLanguageFromLanguageListOptOut	Measure-Command	

New-WSManSessionOption	Remove-SqlAvailabilityRe-	Set-Location
Out-Default	plica	Set-NfsMappedIdentity
Out-File	Remove-TypeData	Set-NfsNetgroup
Out-GridView	Remove-Variable	Set-PSBreakpoint
Out-Host	Remove-WindowsDriver	Set-PSDebug
Out-Null	Remove-WindowsPackage	Set-PSSessionConfigura-
Out-Printer	Remove-WmiObject	tion
Out-String	Remove-WSManInstance	Set-ScheduledJob
Pop-Location	Rename-Computer	Set-ScheduledJobOption
Push-Location	Rename-Item	Set-SecureBootUEFI
Read-Host	Rename-ItemProperty	Set-Service
Receive-DtcDiagnostic-	Repair-WindowsImage	Set-SqlAvailabilityGroup
Transaction	Reset-ComputerMachine-	Set-SqlAvailabilityGrou-
Receive-Job	Password	pListener
Receive-PSSession	Resolve-DnsName	Set-SqlAvailabilityRepli-
Register-CimIndicatio-	Resolve-Path	ca
nEvent	Restart-Computer	Set-SqlHADREndpoint
Register-EngineEvent	Restart-Service	Set-StrictMode
Register-ObjectEvent	Restore-ASDatabase	Set-TpmOwnerAuth
Register-PSSessionConfi-	Restore-Computer	Set-TraceSource
guration	Restore-IscsiVirtualDisk	Set-Variable
Register-ScheduledJob	Restore-SqlDatabase	Set-WheaMemoryPolicy
Register-WmiEvent	Resume-BitsTransfer	Set-WinAcceptLanguage-
Remove-AppxPackage	Resume-Job	FromLanguageListOptOut
Remove-AppxProvisionedPa-	Resume-Service	Set-WinCultureFromLan-
ckage	Resume-SqlAvailabilityDa-	guageListOptOut
Remove-BitsTransfer	tabase	Set-WinDefaultInputMe-
Remove-CertificateEnroll-	Save-Help	thodOverride
mentPolicyServer	Save-WindowsImage	Set-WindowsEdition
Remove-CertificateNotifi-	Select-Object	Set-WindowsProductKey
cationTask	Select-String	Set-WinHomeLocation
Remove-CimInstance	Select-Xml	Set-WinLanguageBarOption
Remove-CimSession	Send-DtcDiagnosticTran-	Set-WinSystemLocale
Remove-Computer	saction	Set-WinUILanguageOverride
Remove-Event	Send-MailMessage	Set-WinUserLanguageList
Remove-EventLog	Set-Acl	Set-WmiInstance
Remove-IscsiServerTarget	Set-Alias	Set-WSManInstance
Remove-IscsiVirtualDisk	Set-AppLockerPolicy	Set-WSManQuickConfig
Remove-IscsiVirtualDiskS-	Set-AuthenticodeSignature	Show-Command
napshot	Set-BitsTransfer	Show-ControlPanelItem
Remove-IscsiVirtualDisk-	Set-BpaResult	Show-EventLog
TargetMapping	Set-CertificateAutoEnroll-	Sort-Object
Remove-Item	mentPolicy	Split-Path
Remove-ItemProperty	Set-CimInstance	Start-BitsTransfer
Remove-Job	Set-Content	Start-DtcDiagnosticRe-
Remove-JobTrigger	Set-Culture	sourceManager
Remove-Module	Set-Date	Start-Job
Remove-NfsMappedIdentity	Set-ExecutionPolicy	Start-Process
Remove-NfsNetgroup	Set-IscsiServerTarget	Start-Service
Remove-PSBreakpoint	Set-IscsiTargetServerSet-	Start-Sleep
Remove-PSDrive	ting	Start-Transaction
Remove-PSSession	Set-IscsiVirtualDisk	Start-Transcript
Remove-PSSnapin	Set-IscsiVirtualDiskSnap-	Stop-Computer
Remove-RoleMember	shot	Stop-DtcDiagnosticResour-
Remove-SqlAvailabilityDa-	Set-Item	ceManager
tabase	Set-ItemProperty	Stop-Job
Remove-SqlAvailability-	Set-JobTrigger	Stop-Process
Group	Set-KdsConfiguration	Stop-Service

Stop-Transcript	Test-PSSessionConfigurationFile	Update-Help
Suspend-BitsTransfer	Test-SqlAvailabilityGroup	Update-List
Suspend-Job	Test-SqlAvailabilityReplica	Update-TypeData
Suspend-Service	Test-SqlDatabaseReplicaState	Use-Transaction
Suspend-SqlAvailabilityDatabase	Test-WSMan	Use-WindowsUnattend
Switch-Certificate	Trace-Command	Wait-Event
Switch-SqlAvailabilityGroup	Unblock-File	Wait-Job
Tee-Object	Unblock-Tpm	Wait-Process
Test-AppLockerPolicy	Undo-DtcDiagnosticTransaction	Where-Object
Test-Certificate	Undo-Transaction	Write-Debug
Test-ComputerSecureChannel	Unregister-Event	Write-Error
Test-Connection	Unregister-PSSessionConfiguration	Write-EventLog
Test-KdsRootKey	Unregister-ScheduledJob	Write-Host
Test-ModuleManifest	Update-FormatData	Write-Output
Test-NfsMappedIdentity		Write-Progress
Test-Path		Write-Verbose
		Write-Warning

B. Les fonctions

A:	Clear-DnsClientCache	Disable-NetAdapterPowerManagement
Add-BCDataCacheExtension	Clear-Host	Disable-NetAdapterQos
Add-BitLockerKeyProtector	Close-SmbOpenFile	Disable-NetAdapterRdma
Add-DnsClientNrptRule	Close-SmbSession	Disable-NetAdapterRsc
Add-DtcClusterTMMapping	Connect-IscsiTarget	Disable-NetAdapterRss
Add-InitiatorIdToMaskingSet	Connect-VirtualDisk	Disable-NetAdapterSriov
Add-NetIPHttpsCertBinding	Copy-NetFirewallRule	Disable-NetAdapterVmq
Add-NetLbfoTeamMember	Copy-NetIPsecMainModeCryptoSet	Disable-NetDnsTransitionConfiguration
Add-NetLbfoTeamNic	Copy-NetIPsecMainModeRule	Disable-NetFirewallRule
Add-NetSwitchTeamMember	Copy-NetIPsecPhase1AuthSet	Disable-NetIPHttpsProfile
Add-OdbcDsn	Copy-NetIPsecPhase2AuthSet	Disable-NetIPsecMainModeRule
Add-PartitionAccessPath	Copy-NetIPsecQuickModeCryptoSet	Disable-NetIPsecRule
Add-PhysicalDisk	Copy-NetIPsecRule	Disable-NetNatTransitionConfiguration
Add-Printer	D:	Disable-OdbcPerfCounter
Add-PrinterDriver	Disable-BC	Disable-PhysicalDiskIndication
Add-PrinterPort	Disable-BCDowngrading	Disable-PSTrace
Add-RDServer	Disable-BCServeOnBattery	Disable-PSWSManCombinedTrace
Add-RDSessionHost	Disable-BitLocker	Disable-RDVirtualDesktoppADMachineAccountReuse
Add-RDVirtualDesktopToCollection	Disable-BitLockerAutoUnlock	Disable-ScheduledTask
Add-TargetPortToMaskingSet	Disable-DAManualEntryPointSelection	Disable-ServerManagerStandardUserRemoting
Add-VirtualDiskToMaskingSet	Disable-MMAgent	Disable-Ual
Add-VpnConnection	Disable-NetAdapter	Disable-WdacBidTrace
B:	Disable-NetAdapterBinding	Disable-WSManTrace
Backup-BitLockerKeyProtector	Disable-NetAdapterChecksumOffload	Disconnect-IscsiTarget
Block-SmbShareAccess	Disable-NetAdapterEncapsulatedPacketTaskOffload	Disconnect-NfsSession
C:	Disable-NetAdapterIPsecOffload	Disconnect-RDUser
cd..	Disable-NetAdapterLso	Disconnect-VirtualDisk
cd\		Dismount-DiskImage
Clear-BCCache		
Clear-BitLockerAutoUnlock		
Clear-Disk		

E:	F:	Get-IscsiTargetPortal
Enable-BCDistributed	Format-Volume	Get-IseSnippet
Enable-BCDowngrading	G:	Get-LogProperties
Enable-BCHostedClient	Get-AppxLastError	Get-MaskingSet
Enable-BCHostedServer	Get-AppxLog	Get-MMAgent
Enable-BCLocal	Get-BCClientConfiguration	Get-NCSIPolicyConfigura-
Enable-BCServeOnBattery	Get-BCContentServerConfi-	tion
Enable-BitLocker	Get-BCDataCache	Get-Net6to4Configuration
Enable-BitLockerAutoUn-	Get-BCDataCacheExtension	Get-NetAdapter
lock	Get-BCHashCache	Get-NetAdapterAdvanced-
Enable-DAManualEntry-	Get-BCHostedCacheServer-	Property
PointSelection	Configuration	Get-NetAdapterBinding
Enable-MMAgent	Get-BCNetworkConfigura-	Get-NetAdapterChecksumOf-
Enable-NetAdapter	tion	fload
Enable-NetAdapterBinding	Get-BCStatus	Get-NetAdapterEncapsula-
Enable-NetAdapterChecksu-	Get-BitLockerVolume	tedPacketTaskOffload
mOffload	Get-ClusteredScheduled-	Get-NetAdapterHardwareIn-
Enable-NetAdapterEncapsu-	Task	fo
latedPacketTaskOffload	Get-CounterSample	Get-NetAdapterIPsecOf-
Enable-NetAdapterIPsecOf-	Get-DAClientExperience-	fload
fload	Configuration	Get-NetAdapterLso
Enable-NetAdapterLso	Get-DAClientExperience-	Get-NetAdapterPowerMana-
Enable-NetAdapterPowerMa-	Configuration	gement
agement	Get-DAClientExperience-	Get-NetAdapterQos
Enable-NetAdapterQos	Get-DAClientExperience-	Get-NetAdapterRdma
Enable-NetAdapterRdma	Get-DAClientExperience-	Get-NetAdapterRsc
Enable-NetAdapterRsc	Get-DAClientExperience-	Get-NetAdapterRss
Enable-NetAdapterRss	Get-DAClientExperience-	Get-NetAdapterSriov
Enable-NetAdapterSriov	Get-DAClientExperience-	Get-NetAdapterSriovVf
Enable-NetAdapterVm	Get-DAClientExperience-	Get-NetAdapterStatistics
q	Get-DAClientExperience-	Get-NetAdapterVm
Enable-NetDnsTransition-	Get-DAClientExperience-	q
Configuration	Get-DAClientExperience-	Get-NetAdapterVmQueue
Enable-NetFirewallRule	Get-DAClientExperience-	Get-NetAdapterVPort
Enable-NetIPHttpsProfile	Get-DAClientExperience-	Get-NetConnectionProfile
Enable-NetIPsecMainMode-	Get-DAClientExperience-	Get-NetDnsTransitionCon-
Rule	Get-DAClientExperience-	figuration
Enable-NetIPsecRule	Get-DAClientExperience-	Get-NetDnsTransitionMo-
Enable-NetNatTransition-	Get-DAClientExperience-	onitoring
Configuration	Get-DAClientExperience-	Get-NetFirewallAddress-
Enable-OdbcPerfCounter	Get-DAClientExperience-	Filter
Enable-PhysicalDiskIndi-	Get-DAClientExperience-	Get-NetFirewallApplica-
cation	Get-DAClientExperience-	tionFilter
Enable-PSTrace	Get-DAClientExperience-	Get-NetFirewallInterface-
Enable-PSWSManCombined-	Get-DAClientExperience-	Filter
Trace	Get-DAClientExperience-	Get-NetFirewallInterface-
Enable-RDVirtualDesкто-	Get-DAClientExperience-	TypeFilter
pADMachineAccountReuse	Get-DAClientExperience-	Get-NetFirewallPortFilter
Enable-ScheduledTask	Get-DAClientExperience-	Get-NetFirewallProfile
Enable-ServerManagerStan-	Get-DAClientExperience-	Get-NetFirewallRule
dardUserRemoting	Get-DAClientExperience-	Get-NetFirewallSecurity-
Enable-Ual	Get-DAClientExperience-	Filter
Enable-WdacBidTrace	Get-DAClientExperience-	Get-NetFirewallService-
Enable-WSManTrace	Get-DAClientExperience-	Filter
Export-BCCachePackage	Get-DAClientExperience-	Get-NetFirewallSetting
Export-BCSecretKey	Get-DAClientExperience-	Get-NetIPAddress
Export-RDPersonalVirtual-	Get-DAClientExperience-	Get-NetIPConfiguration
DesktopAssignment	Get-DAClientExperience-	Get-NetIPHttpsConfigura-
Export-ScheduledTask	Get-DAClientExperience-	tion

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

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

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

Set-NetLbfoTeam	Set-RDRemoteApp	Stop-RDVirtualDesktopCollectionJob
Set-NetLbfoTeamMember	Set-RDRemoteDesktop	Stop-ScheduledTask
Set-NetLbfoTeamNic	Set-RDSessionCollectionConfiguration	Stop-Trace
Set-NetNatTransitionConfiguration	Set-RDSessionHost	Suspend-BitLocker
Set-NetNeighbor	Set-RDVirtualDesktopCollectionConfiguration	Suspend-PrintJob
Set-NetOffloadGlobalSetting	Set-RDVirtualDesktopConcurrency	Sync-NetIPsecRule
Set-NetQosPolicy	Set-RDVirtualDesktopIdleCount	T:
Set-NetRoute	Set-RDVirtualDesktopTemplateExportPath	TabExpansion2
Set-NetTCPSetting	Set-RDWorkspace	Test-Dtc
Set-NetTeredoConfiguration	Set-ResiliencySetting	Test-NfsMappingStore
Set-NetUDPSetting	Set-ScheduledTask	Test-RDOUAccess
Set-NfsClientConfiguration	Set-SmbClientConfiguration	Test-RDVirtualDesktopAD-MachineAccountReuse
Set-NfsClientgroup	Set-SmbServerConfiguration	U:
Set-NfsMappingStore	Set-SmbShare	Unblock-SmbShareAccess
Set-NfsNetgroupStore	Set-StoragePool	Uninstall-Dtc
Set-NfsServerConfiguration	Set-StorageSetting	Uninstall-WindowsFeature
Set-NfsShare	Set-StorageSubSystem	Unlock-BitLocker
Set-OdbcDriver	Set-VirtualDisk	Unregister-ClusteredScheduledTask
Set-OdbcDsn	Set-Volume	Unregister-IscsiSession
Set-Partition	Set-VolumeScrubPolicy	Unregister-ScheduledTask
Set-PhysicalDisk	Set-VpnConnection	Unregister-WindowsDeveloperLicense
Set-PrintConfiguration	Set-VpnConnectionProxy	Update-Disk
Set-Printer	Show-NetFirewallRule	Update-HostStorageCache
Set-PrinterProperty	Show-NetIPsecRule	Update-IscsiTarget
Set-RDActiveManagementServer	Show-VirtualDisk	Update-IscsiTargetPortal
Set-RDCertificate	Show-WindowsDeveloperLicenseRegistration	Update-NetIPsecRule
Set-RDClientAccessName	Start-Dtc	Update-RDVirtualDesktopCollection
Set-RDConnectionBrokerHighAvailability	Start-DtcTransactionsTraceSession	Update-SmbMultichannel-Connection
Set-RDDatabaseConnectionString	Start-PerformanceCollector	Update-StorageProvider-Cache
Set-RDDeploymentGatewayConfiguration	Start-ScheduledTask	V:
Set-RDFileTypeAssociation	Start-Trace	W:
Set-RDLicenseConfiguration	Stop-Dtc	Write-DtcTransactionsTraceSession
Set-RDPersonalVirtualDesktopAssignment	Stop-DtcTransactionsTraceSession	X:
Set-RDPersonalVirtualDesktopPatchSchedule	Stop-PerformanceCollector	Y:
		Z:

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

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

DatePart	<div>\$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</div>
DateSerial	<div>MyDate1 = DateSerial(2006, 12, 31) \$a = get-date -y 2006 -mo 12 -day 31</div>
DateValue	<div>\$a = [datetime] "12/1/2006"</div>
Day	<div>\$a = (get-date).day</div>
Eval	<div>\$a = 2 + 2 -eq 45</div>
Exp	<div>\$a = [math]::exp(2)</div>
Filter	<div>\$a = "Monday","Month","Merry","Mansion","Modest" \$b = (\$a where-object {\$_ -like "Mon*"})</div>
FormatCurrency	<div>\$a = 1000 \$a = "{0:C}" -f \$a</div>
FormatDateTime	<div>\$a = (get-date).tolongdatestring() \$a = (get-date).toshortdatestring() \$a = (get-date).tolongtimestring() \$a = (get-date).toshorttimestring()</div>
FormatNumber	<div>\$a = 11 \$a = "{0:N6}" -f \$a</div>
FormatPercent	<div>\$a = .113 \$a = "{0:P1}" -f \$a</div>
GetLocale	<div>\$a = (get-culture).lcid \$a = (get-culture).displayname</div>
Hex	<div>\$a = 4517 \$a = "{0:X}" -f \$a</div>
Hour	<div>\$a = (get-date).hour</div>
InputBox	<div>\$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")</div>
InStr	<div>\$a = "wombat" \$b = \$a.contains("m") \$b = \$a.indexof("m")</div>
InStrRev	<div>\$a = "1234x6789x1234" \$b = \$a.lastindexofany("x")</div>
Int/Fix	<div>\$a = 11.98 \$a = [math]::truncate(\$a)</div>
IsArray	<div>\$a = 22,5,10,8,12,9,80 \$b = \$a -is [array]</div>
IsDate	<div>\$a = 11/2/2006 \$a -is [datetime] \$a = [datetime] "11/2/2006"</div>
IsEmpty	<div>\$a = "" \$b = \$a.length -eq 0</div>
IsNull	<div>\$a = \$z -eq \$null</div>
IsNumeric	<div>\$a = 44.5</div>

	[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 = "ABCDEFGHJKLMNOPQRSTUVWXYZ" \$a = \$a.ToLower()
Left	\$a="ABCDEFGHJKLMNOPQRSTUVWXYZ" \$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="ABCDEFGH" \$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 = "bxnxbx" \$a = \$a -replace("x","a")
RGB	\$blue = 10 \$green = 10 \$red = 10 \$a = [long] (\$blue + (\$green * 256) + (\$red * 65536))
Right	\$a = "ABCDEFGHJKLMNOPQRSTUVWXYZ" \$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