Table des matières

I. Présentation

A. Préambule

Ce document est un support de cours dont l'objet est de fournir les clés de compréhension du PowerShell. Il ne peut pas faire l'objet de reproductions à des fins commerciales sans le consentement express 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 : le Korne Shell, le Bourne Shell, le Bash, le C-Shell, le Z-Shell. Avec Dos, puis Windows, Microsoft a développé différentes technologies de scripting. Initialement, il y a eu les commandes autour du DOS. Sous Windows NT, nous avions eu droit à Kix. Avec Windows, Bill Gates voulait faire de Visual Basic le langage universel. Avant le PowerShell, il y eut Vbscript, utilisé avec Windows Scripting Host. Et puis, avec l'avènement de .Net, Microsoft a décidé de mettre en avant le PowerShell. Certains autres langages tels que Perl, Python, Php-Cli sont eux aussi très utilisés et 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 senteur de l'exécution due à l'utilisation du Framework .Net.
PowerShell s'exécute aujourd'hui sur Linux.

C. PowerShell 5

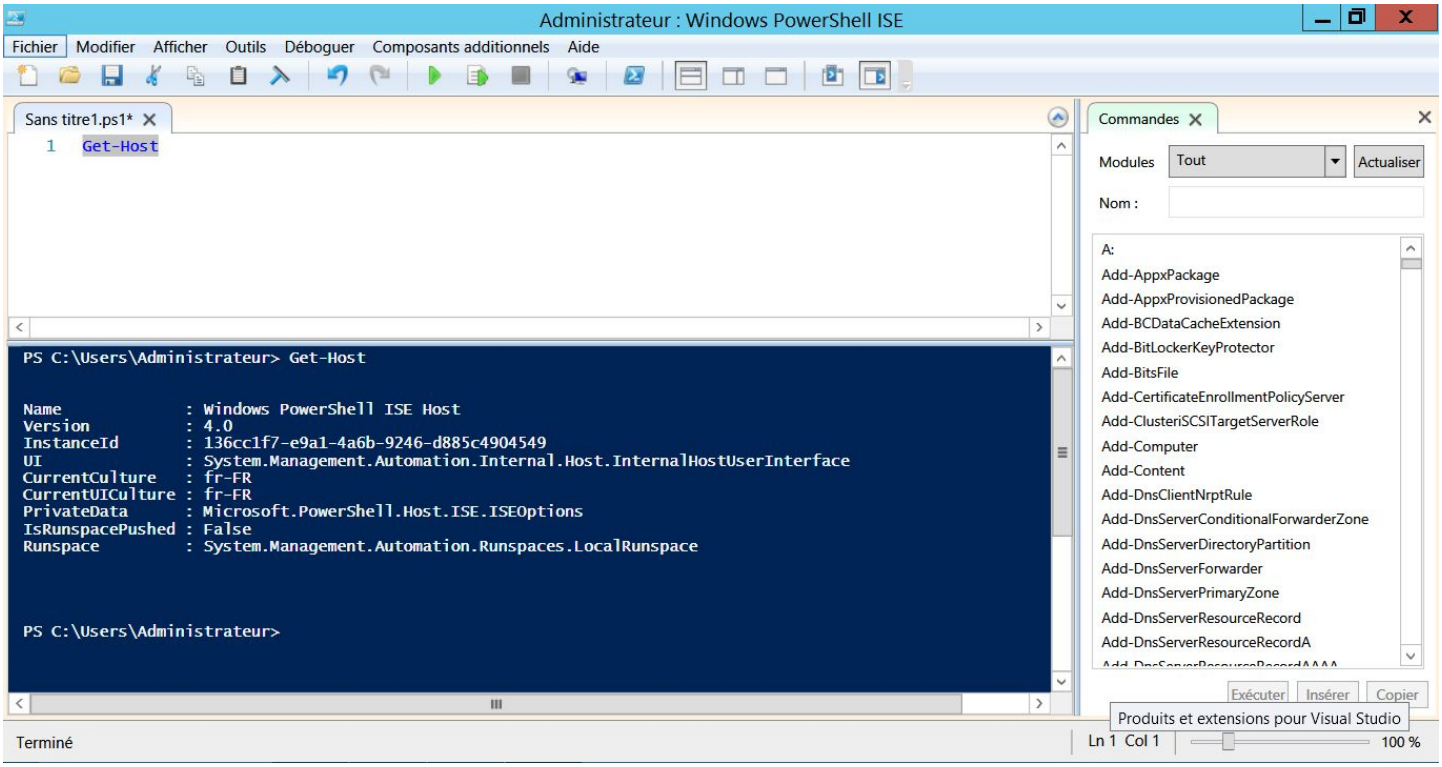
Windows PowerShell 5.0 nécessite Microsoft .NET Framework 4.5 + et WMF (Windows Management Framework) 5.0+. La nouvelle version de PowerShell est native sur Windows 8.1 et Windows Server 2012 R2.
Pour déterminer la version de votre Powershell :

```
Get-Host | Select-Object Version
(Get-Host).Version
[string](Get-Host).Version
[string](Get-Host).Version.Major+'.'+[string](Get-Host).Version.Minor
(Get-Host).Version.Major.ToString()+'.'+(Get-Host).Version.Minor.ToString()
```

D. Les outils

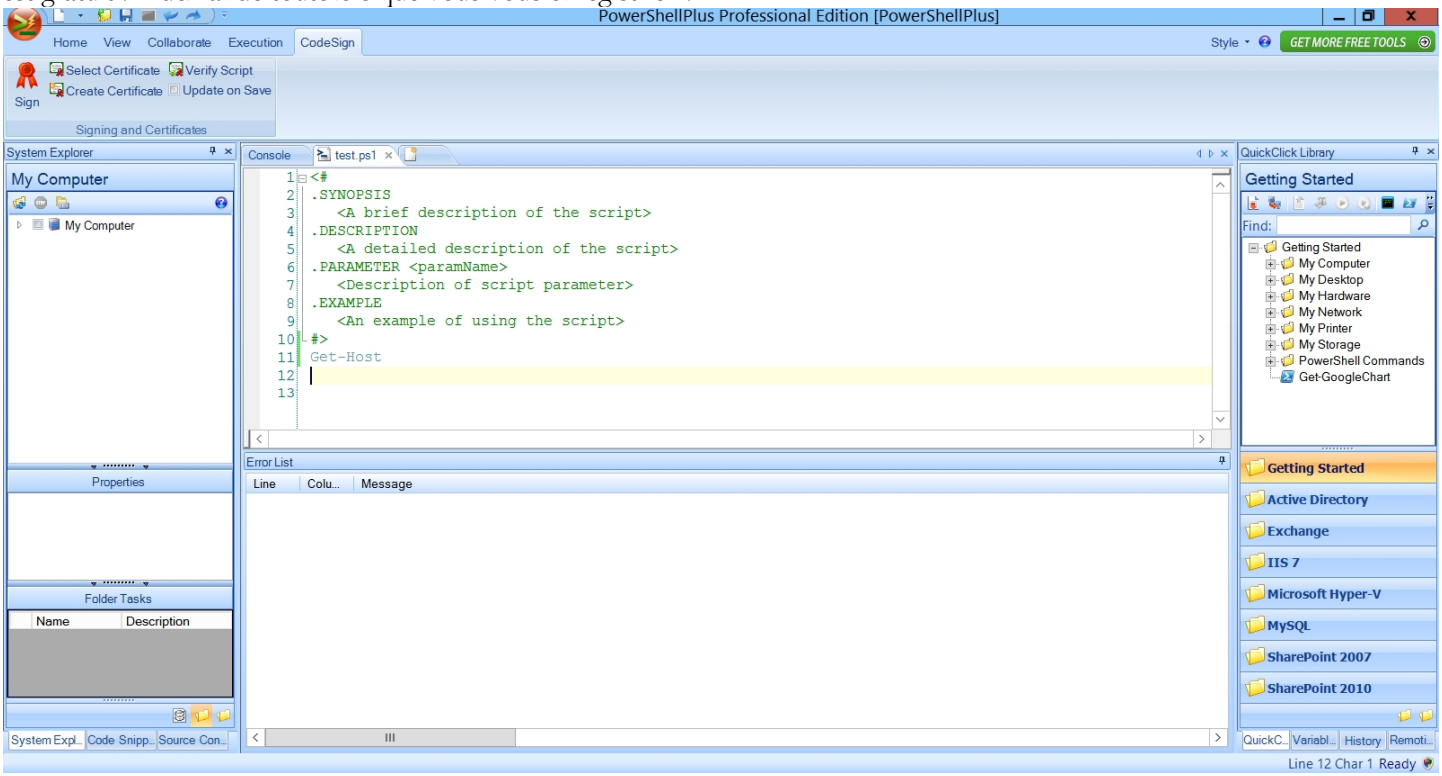
1. Windows PowerShell ISE, intégré à Windows

Pour créer des scripts en PowerShell, vous pouvez utiliser l'éditeur intégré à Windows, Windows PowerShell ISE. Il est stocké dans le dossier C:\Windows\System32\WindowsPowerShell\v1.0, malgré que l'on soit passé à la version PowerShell 5.0.



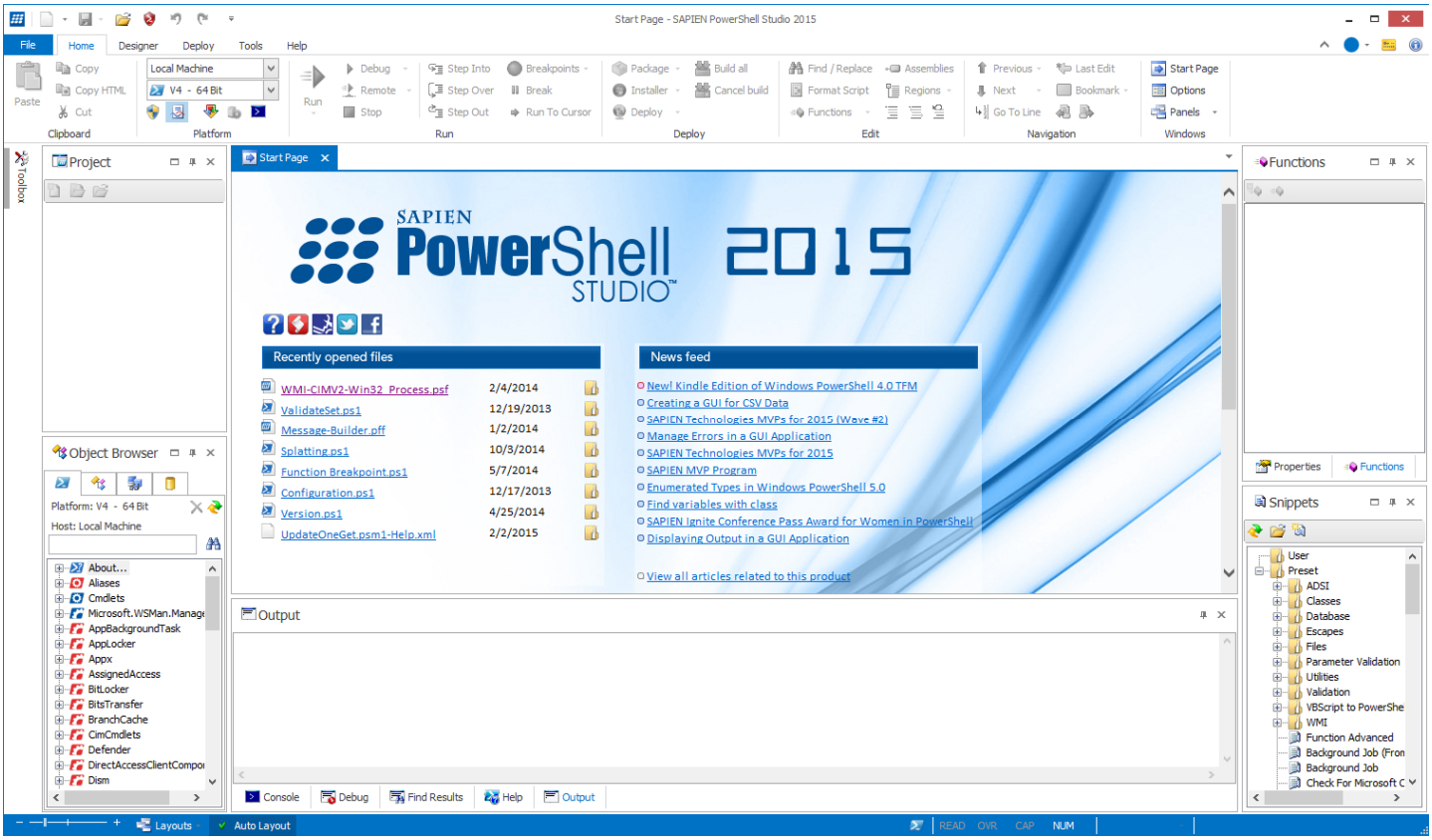
2. Idera PowerShell Plus

Idera PowerShell Plus C'est, sans doute, le plus bel outil de sa catégorie, malgré l'affichage de la console dans un onglet. Et il est gratuit ! Il demande toutefois que vous vous enregistriez !



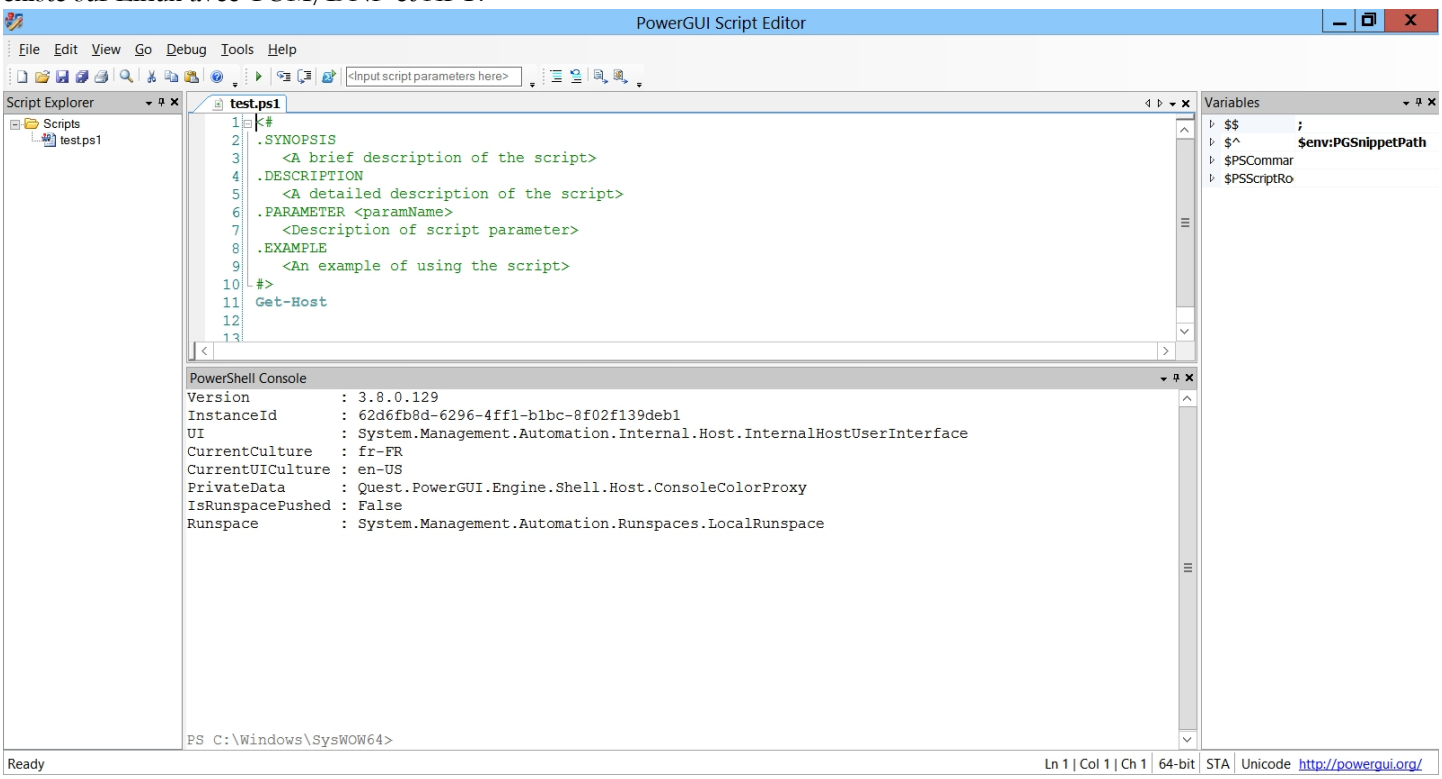
3. Sapien PowerShell Studio 2019

Sapien PowerShell Studio est un outil payant que vous pouvez essayer 45 jours.



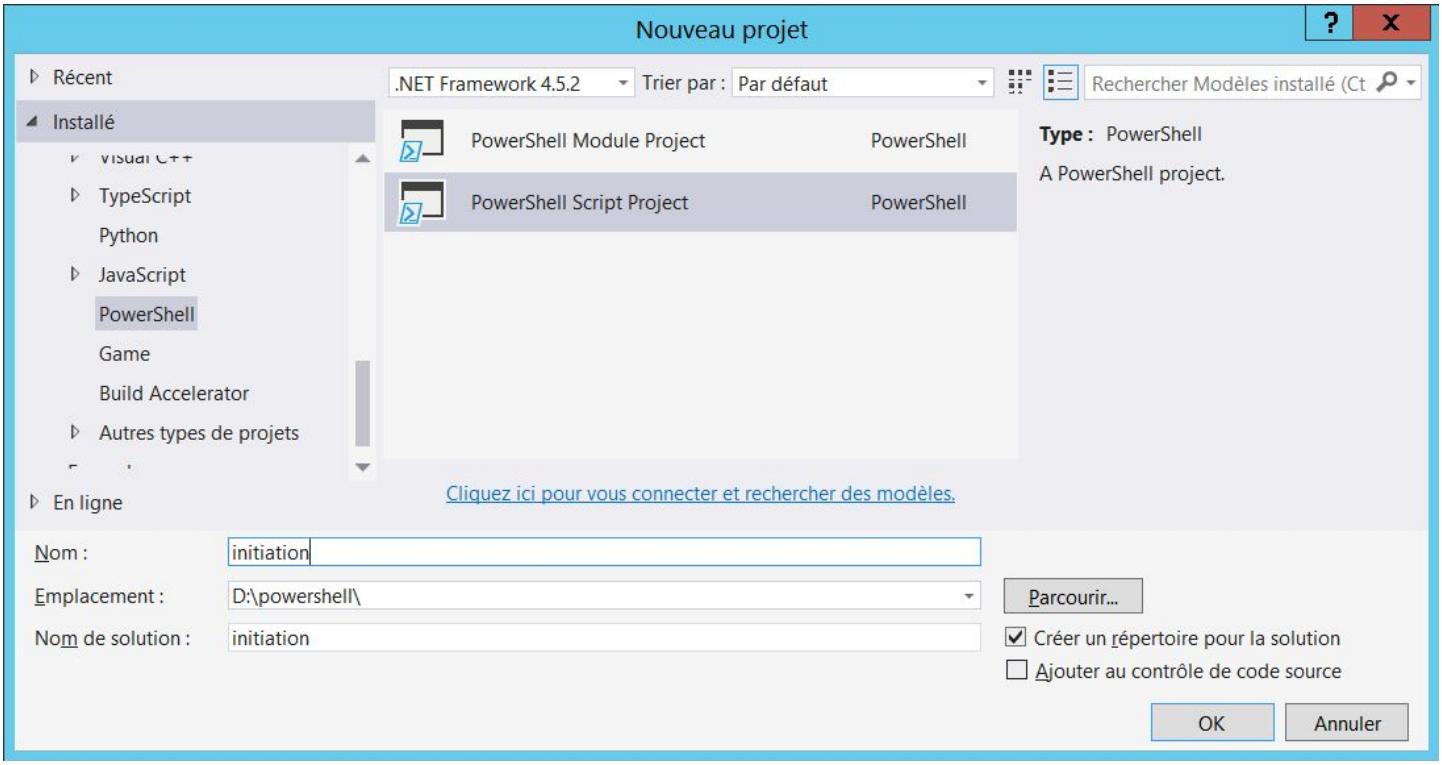
4. PowerGUI

L'éditeur gratuit de chez Quest (Dell) se présente comme celui de Microsoft. L'intérêt de PowerGui est la communauté qui lui est rattachée ! Vous pouvez l'installer à l'aide de Chocolatey, un gestionnaire de paquets sous Windows, à la manière de ce qui existe sur Linux avec YUM/DNF et APT.

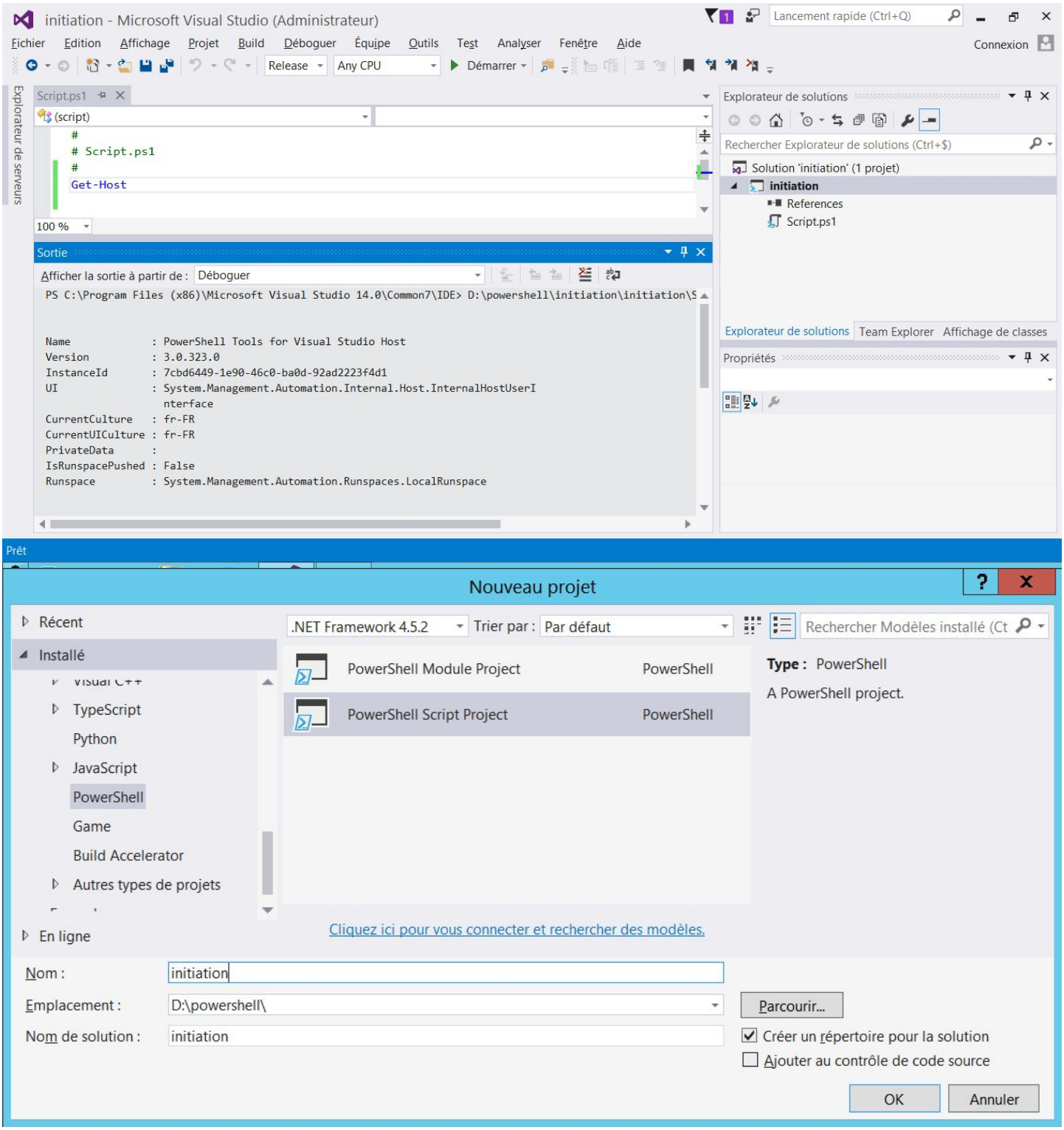


5. Visual Studio Community Edition 2019

Pour créer un nouveau projet, allez dans *Fichier -> Nouveau Projet* :



Après avoir tapé votre code, exécutez le en cliquant sur Démarrer :


initiation - Microsoft Visual Studio (Administrateur)
Fichier Edition Affichage Projet Build Déboguer Équipe Outils Test Analyser Fenêtre Aide
Connexion
Release
Any CPU
Démarrer
Script.ps1
(script)
Script.ps1
Get-Host
100 %
Sortie
Afficher la sortie à partir de : Déboguer
PS C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\IDE> D:\powershell\initiation\initiation\S
Name : PowerShell Tools for Visual Studio Host
Version : 3.0.323.0
InstanceId : 7cbd6449-1e90-46c0-ba0d-92ad2223f4d1
UI : System.Management.Automation.Internal.Host.InternalHostUserI
nterface
CurrentCulture : fr-FR
CurrentUICulture : fr-FR
PrivateData :
IsRunspacePushed : False
Runspace : System.Management.Automation.Runspaces.LocalRunspace
Explorateur de solutions
Solution 'initiation' (1 projet)
initiation
References
Script.ps1
Explorateur de solutions Team Explorer Affichage de classes
Propriétés
Prêt
Nouveau projet
Récent
Installé
TypeScript
Python
JavaScript
PowerShell
Game
Build Accelerator
Autres types de projets
En ligne
.NET Framework 4.5.2
Trier par : Par défaut
PowerShell Module Project PowerShell
PowerShell Script Project PowerShell
Type : PowerShell
A PowerShell project.
Cliquez ici pour vous connecter et rechercher des modèles.
Nom : initiation
Emplacement : D:\powershell\
Nom de solution : initiation
Parcourir...
Créer un répertoire pour la solution
Ajouter au contrôle de code source
OK
Annuler

II. Premiers pas

E. Les applets de commande ou cmdlets

Le langage PowerShell s'appuie sur un jeu de commandes qui peut être enrichi par l'installation de rôles sur les serveurs ou bien de logiciels comme Microsoft Exchange, Microsoft SQL Server. Elle se compose généralement d'un verbe suivi d'un mot, comme Get-Host, Set-Content, etc.

F. L'interpréteur

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

G. 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 autorisé
Unrestricted	Aucune limite pour l'exécution des scripts
Undefined	Indéfini

```
Get-ExecutionPolicy -List
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned -Force
Set-ExecutionPolicy -Scope LocalMachine -ExecutionPolicy RemoteSigned -Force
Set-ExecutionPolicy -Scope Process -ExecutionPolicy RemoteSigned -Force
```

3. Signature d'un exécutable, d'une DLL, d'un driver

```
Get-AuthenticodeSignature "C:\windows\notepad.exe"
Get-AuthenticodeSignature "C:\windows\system32\kernel32.dll"
Get-AuthenticodeSignature "C:\windows\system32\drivers\acpi.sys"
```

4. Voir aussi

Get-Help about_Execution_Policies
Get-Help about_Profiles
Get-ExecutionPolicy
Set-ExecutionPolicy
Set-AuthenticodeSignature

5. 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

6. Associer un certificate à un script

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

7. Certificat auto-signé via OpenSSL

Vous pouvez installer OpenSSL sous Windows afin de générrer très simplement votre certificat.

https://www.dsfc.net/infrastructure/securite/creer-un-certificat-auto-signe-avec-openssl-sous-windows/

H. Aide

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

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

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

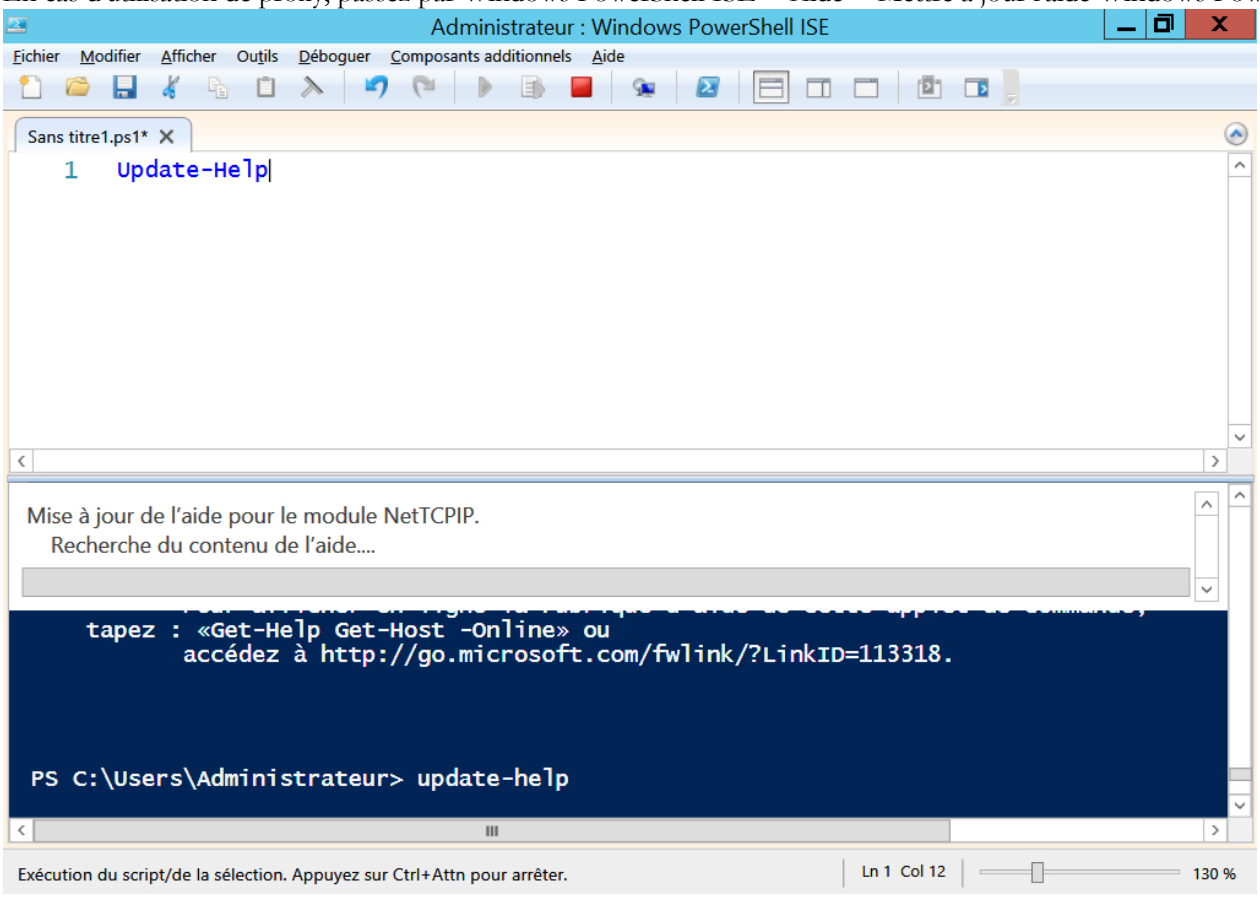
```
Get-Command -Verb Get
Get-Command -Module NetTcpIp
Get-Command *wmi*
```

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-Help * -Parameter ComputerName
Get-Help 'Update-Help' -ShowWindow
```

4. Actualiser l'aide

A faire impérativement pour les développeurs de scripts PowerShell :
`Update-Help`
En cas d'utilisation de proxy, passez par Windows PowerShell ISE > Aide > Mettre à jour l'aide Windows PowerShell.



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
Get-EventLog -logname system -Newest 1| format-list -property *
```

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
(Get-History).CommandLine
```

11. Les alias

En terme de performance, l'usage des alias n'est pas recommandé !

a) Liste des alias

La commande Alias vous permet d'obtenir tous les alias de commandes définis dans votre environnement.

```
write -> Write-Output
wjb -> Wait-Job
where -> Where-Object
wget -> Invoke-WebRequest
type -> Get-Content
trcm -> Trace-Command
tee -> Tee-Object
swmi -> Set-WmiInstance
sv -> Set-Variable
sujb -> Suspend-Job
```

```
stz -> Set-TimeZone
start -> Start-Process
spsv -> Stop-Service
spps -> Stop-Process
spjb -> Stop-Job
sp -> Set-ItemProperty
sort -> Sort-Object
sls -> Select-String
sleep -> Start-Sleep
sl -> Set-Location
si -> Set-Item
shcm -> Show-Command
Set-DnsServerRRL
set -> Set-Variable
select -> Select-Object
scb -> Set-Clipboard
sc -> Set-Content
sbp -> Set-PSBreakpoint
sasv -> Start-Service
saps -> Start-Process
sal -> Set-Alias
sajb -> Start-Job
rwmi -> Remove-WmiObject
rvpa -> Resolve-Path
rv -> Remove-Variable
rujb -> Resume-Job
rsnp -> Remove-PSSnapin
rsn -> Remove-PSSession
rp -> Remove-ItemProperty
rnp -> Rename-ItemProperty
rni -> Rename-Item
rmo -> Remove-Module
rmdir -> Remove-Item
rm -> Remove-Item
rjb -> Remove-Job
ri -> Remove-Item
ren -> Rename-Item
rdr -> Remove-PSDrive
rd -> Remove-Item
rcsn -> Receive-PSSession
rcjb -> Receive-Job
rbp -> Remove-PSBreakpoint
r -> Invoke-History
pwd -> Get-Location
pushd -> Push-Location
ps -> Get-Process
popd -> Pop-Location
oh -> Out-Host
ogv -> Out-GridView
nv -> New-Variable
nsn -> New-PSSession
npssc -> New-PSSessionConfigurationFile
nmo -> New-Module
ni -> New-Item
ndr -> New-PSDrive
nal -> New-Alias
mv -> Move-Item
mp -> Move-ItemProperty
move -> Move-Item
mount -> New-PSDrive
mi -> Move-Item
measure -> Measure-Object
```

```
md -> mkdir
man -> help
ls -> Get-ChildItem
lp -> Out-Printer
kill -> Stop-Process
iwr -> Invoke-WebRequest
iwmi -> Invoke-WmiMethod
ise -> powershell_ise.exe
irm -> Invoke-RestMethod
ipsn -> Import-PSSession
ipmo -> Import-Module
ipcsv -> Import-Csv
ipal -> Import-Alias
ii -> Invoke-Item
ihy -> Invoke-History
iex -> Invoke-Expression
icm -> Invoke-Command
history -> Get-History
h -> Get-History
gwmi -> Get-WmiObject
gv -> Get-Variable
gu -> Get-Unique
gtz -> Get-TimeZone
gsv -> Get-Service
gsnp -> Get-PSSnapin
gsn -> Get-PSSession
group -> Group-Object
gpv -> Get-ItemPropertyValue
gps -> Get-Process
gp -> Get-ItemProperty
gmo -> Get-Module
gm -> Get-Member
gl -> Get-Location
gjb -> Get-Job
gin -> Get-ComputerInfo
gi -> Get-Item
ghy -> Get-History
Get-DnsServerRRL
gdr -> Get-PSDrive
gcs -> Get-PSCallStack
gcm -> Get-Command
gci -> Get-ChildItem
gcb -> Get-Clipboard
gc -> Get-Content
gbp -> Get-PSBreakpoint
gal -> Get-Alias
fw -> Format-Wide
ft -> Format-Table
foreach -> ForEach-Object
fl -> Format-List
fhx -> Format-Hex
fc -> Format-Custom
exsn -> Exit-PSSession
Export-DnsServerTrustAnchor
etsn -> Enter-PSSession
erase -> Remove-Item
epsn -> Export-PSSession
epcsv -> Export-Csv
epal -> Export-Alias
echo -> Write-Output
ebp -> Enable-PSBreakpoint
dnsn -> Disconnect-PSSession
```

```
dir -> Get-ChildItem
diff -> Compare-Object
del -> Remove-Item
dbp -> Disable-PSBreakpoint
cvpa -> Convert-Path
curl -> Invoke-WebRequest
cpp -> Copy-ItemProperty
cpi -> Copy-Item
cp -> Copy-Item
copy -> Copy-Item
compare -> Compare-Object
cnsn -> Connect-PSSession
clv -> Clear-Variable
cls -> Clear-Host
clp -> Clear-ItemProperty
cli -> Clear-Item
clhy -> Clear-History
clear -> Clear-Host
clc -> Clear-Content
chdir -> Set-Location
CFS -> ConvertFrom-String
cd -> Set-Location
cat -> Get-Content
asnp -> Add-PSSnapin
ac -> Add-Content
? -> Where-Object
% -> ForEach-Object
```

b) Détruire un alias

```
Remove-Item alias:dir
```

c) Création d'alias

```
Set-Alias Show Get-ChildItem
Show
Remove-Item alias:Show
```

I. Exécution des scripts

1. Exécution d'un script

Vous pouvez taper le chemin complet du script.

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

Dans le répertoire courant, tapez à partir de la ligne de commande.

```
powershell .\monscript.ps1
powershell ./monscript.ps1
```

2. « Compilation » d'un script avec PS2EXE

Il est erroné de parler de compilation concernant l'environnement .Net. En fait, il s'agit du passage à un format ByteCode compressé.

a) Utilisation de la commande ps2exe.ps1 :

```
powershell.exe -command "&'.\ps2exe.ps1' [-inputFile] '<file_name>' [-outputFile] '<file_name>'
               [-verbose] [-debug] [-runtime20|-runtime40] [-lcid <id>] [-x86|-x64] [-Sta|-Mta]
               [-noConsole] [-iconFile '<file_name>']"

   inputFile = Powershell script that you want to convert to EXE
  outputFile = destination EXE file name
```

```
     verbose = output verbose informations - if any
       debug = generate debug informations for output file
   runtime20 = this switch forces PS2EXE to create a config file for the generated EXE that
contains the
               "supported .NET Framework versions" setting for .NET Framework 2.0/3.x for
PowerShell 2.0
   runtime40 = this switch forces PS2EXE to create a config file for the generated EXE that
contains the
               "supported .NET Framework versions" setting for .NET Framework 4.x for PowerShell
3.0 or higher
        lcid = location ID for the compiled EXE. Current user culture if not specified
         x86 = compile for 32-bit runtime only
         x64 = compile for 64-bit runtime only
         sta = Single Thread Apartment Mode
         mta = Multi Thread Apartment Mode
   noConsole = the resulting EXE file will be a Windows Forms app without a console window
    iconFile = icon file name for the compiled EXE
       title = title information (displayed in details tab of Windows Explorer's properties
dialog)
 description = description information (not displayed, but embedded in executable)
     company = company information (not displayed, but embedded in executable)
     product = product information (displayed in details tab of Windows Explorer's properties
dialog)
   copyright = copyright information (displayed in details tab of Windows Explorer's properties
dialog)
   trademark = trademark information (displayed in details tab of Windows Explorer's properties
dialog)
     version = version information (displayed in details tab of Windows Explorer's properties
dialog)
requireAdmin = if UAC is enabled, compiled EXE run only in elevated context (UAC dialog appears
if required)
  virtualize = application virtualization is activated (forcing x86 runtime)
```

b) Exemples

```
powershell.exe -command "&'.\ps2exe.ps1' -inputFile 'version-powershell.ps1' -outputfile
'version-powershell.exe' -runtime50"
powershell.exe ./ps2exe.ps1 -inputFile version-powershell.ps1 -outputfile version-powershell.exe
-runtime51
```

c) Source :

https://www.dsfc.net/developpement/scripting/compiler-powershell-ps2exe/

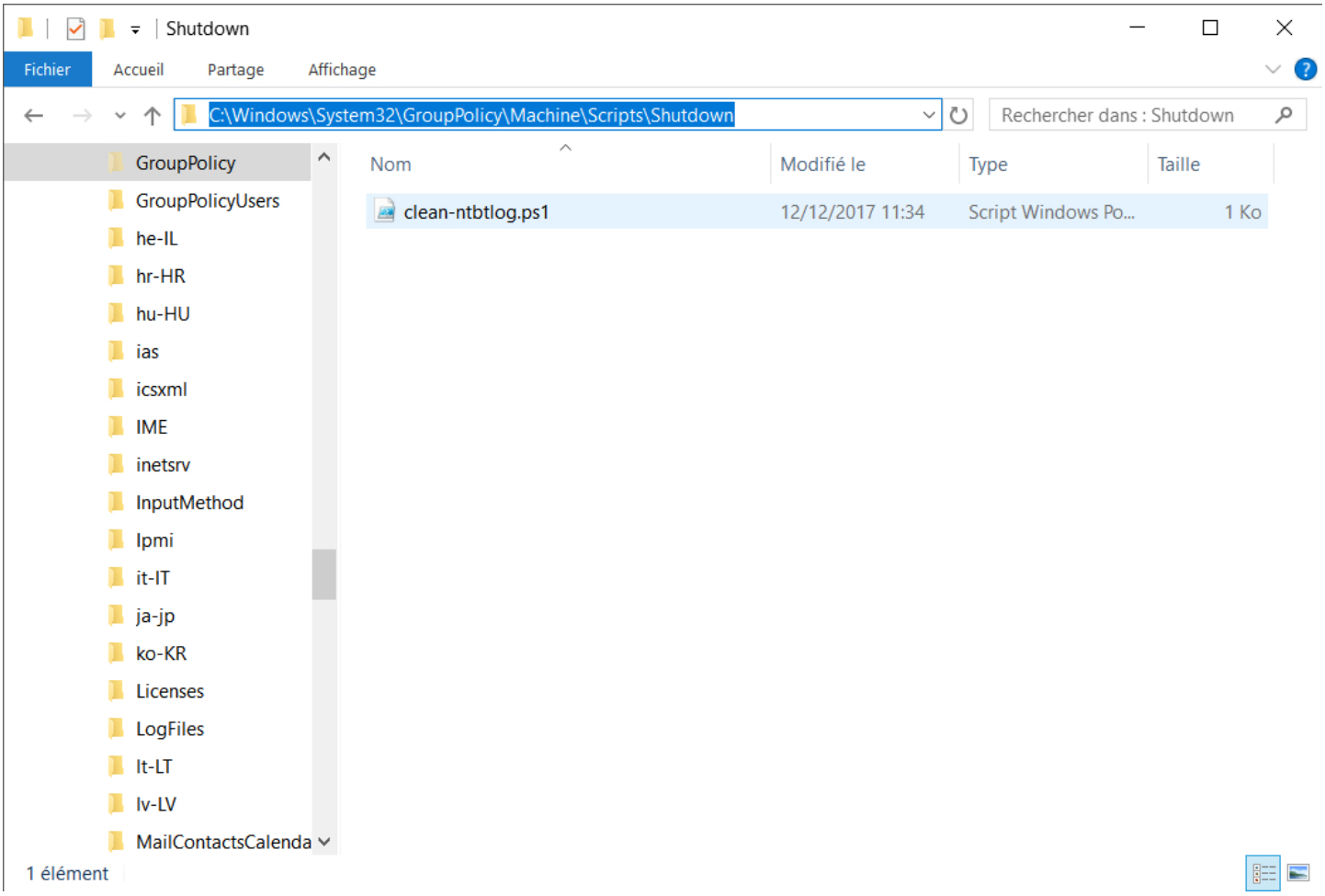
d) Alternative : Ps1 To Exe

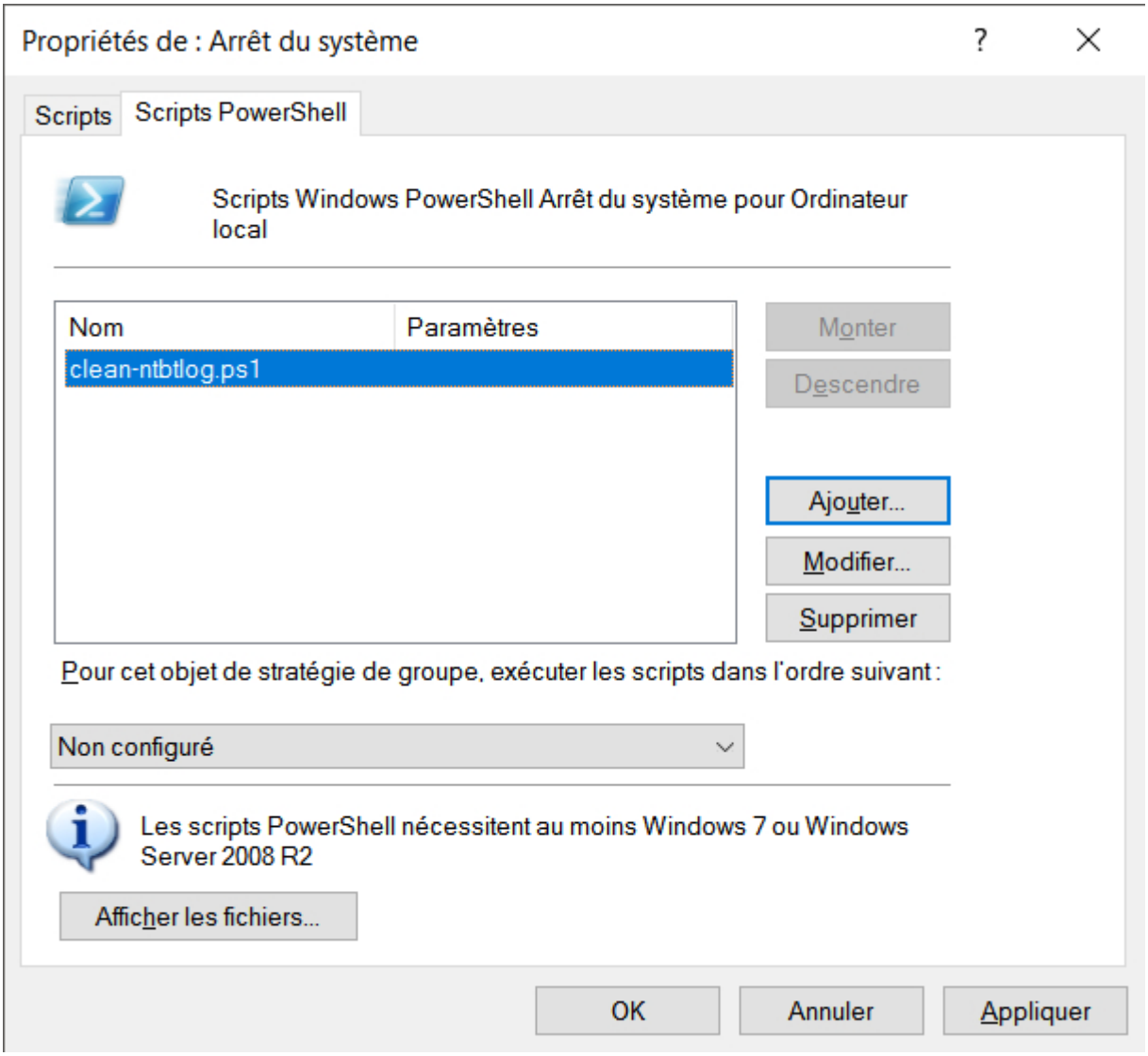
Il s'agit d'un autre "compilateur".
Source : https://www.dsfc.net/informatique/ps1-to-exe-compilateur-powershell/

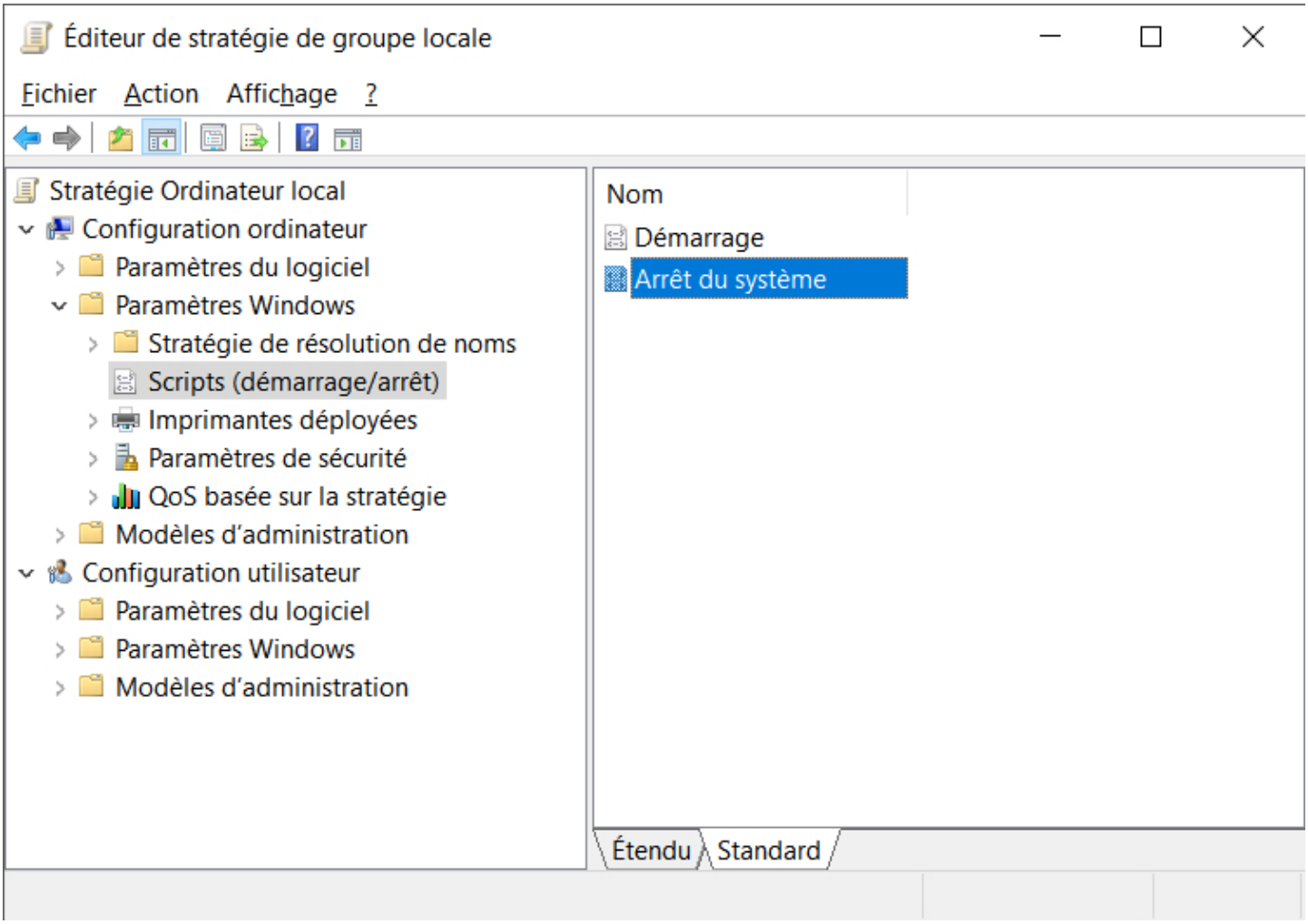
3. Appel d'un autre script

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

4. Script de démarrage, d'arrêt, de connexion, de déconnexion




Propriétés de : Arrêt du système
Scripts
Scripts PowerShell
Scripts Windows PowerShell Arrêt du système pour Ordinateur local
Nom
Paramètres
clean-ntbtlog.ps1
Monter
Descendre
Ajouter...
Modifier...
Supprimer
Pour cet objet de stratégie de groupe, exécuter les scripts dans l'ordre suivant :
Non configuré
Les scripts PowerShell nécessitent au moins Windows 7 ou Windows Server 2008 R2
Afficher les fichiers...
OK
Annuler
Appliquer



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

a) ping

```
#Test de connectivité avec le résultat de la commande Windows ping.exe
$ping=& {c:\windows\system32\ping.exe -n 1 -w 1000 www.yahoo.fr}
$ping

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]
}
```

b) Mac Address

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

```
        break
    }
}
```

6. Variable d'environnement

```
Foreach($item in (Get-ChildItem env:\))
{
    "$($item.Key) : $($item.value)"
}
Get-ChildItem env:\|ForEach{
    $_.key+':'+$_.value
}

$env:COMPUTERNAME
```

7. Ouvrir un programme, un document

```
Invoke-Item 'c:\windows\system32\calc.exe'
invoke-item 'd:\opensearch-mycroft.txt'
```

8. 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}
```

9. Tempo

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

10. 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
```

11. Envoi de mail

a) Méthode Send-MailMessage

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

b) 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)
```

J. Historique

1. Visualiser l'historique

```
Get-History
Get-History 32 -count 32
$MaximumHistoryCount = 32767
```

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
```

K. Informations de langue

```
Get-Culture
Get-UICulture
Set-Culture fr-FR
```

L. 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"
```

M. Commentaires

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

N. Instruction sur plusieurs lignes

Le caratère ` (AltGr 7) permet de prolonger une instruction sur plusieurs lignes.

II. Cmdlets système

A. Le journal d'événements

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

Pour les démarrages de la machine :

```
Get-EventLog -LogName System|Where-Object {$_.Source -like "*Kernel-General*" -and $_.EventID -eq
12}|Select-Object TimeGenerated
 (Get-EventLog -LogName System|Where-Object {$_.Source -like "*Kernel-General*" -and $_.EventID -
eq 12}|Measure-Object).count
Pour les arrêts de la machine :
Get-EventLog -logname system|Where-Object {$_.Source -LIKE '*Kernel-General*' -AND $_.EventID -EQ
13}|Select-Object Source,EventID,TimeWritten

Get-EventLog -LogName Application | Where EntryType -EQ "Error"
Get-EventLog -LogName System | Where {$_.EntryType -EQ "Error" -AND $_.Source -EQ "volmgr"}|
Select Message

Get-WinEvent -ListLog *|Where LogName -Match 'Policy'
Get-WinEvent -LogName 'Microsoft-Windows-GroupPolicy/Operational'|Where LevelDisplayName -EQ
'Erreur'

#Liste des journaux comprenant des entrées
Get-WinEvent -ListLog  * | Where RecordCount -GT 0

Get-EventLog -LogName Application -After '10/12/2018 10:00:00'| Where-Object EntryType -EQ
"Error"|Select-Object -Property TimeWritten,Message|Out-GridView

#Erreur d'ouverture de session
Get-EventLog -LogName security|Where InstanceID -EQ 4776|Select TimeGenerated,
UserName,MachineName,Message|Out-GridView
```

B. Les services (illustration du pipelining)

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."
```

5. Redémarrage du spooler

```
Clear
Stop-Service -Name Spooler
Remove-Item C:\windows\system32\spool\Printers\*.* -Force
Start-Service -Name Spooler
```

6. Désactivation d'une liste de services

```
Clear
Get-Service|Select Name|ForEach {
    If ($_.Name -in ('AdobeARMservice','Areuhhh'))
    {
        Stop-Service -Name $_.Name
        Set-Service -Name $_.Name -startuptype Disabled
    }
}
```

7. Activation de services

```
Clear
Get-Service|Select Name|ForEach {
    If ($_.Name -in 'sppsvc','W32Time','RemoteRegistry')
    {
        Set-Service -Name $_.Name -startuptype Automatic
    }
    Start-Service -Name $_.Name
}
```

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 { $_.starttime.minute -lt 30} | select name, starttime
Get-Process|Get-Member|Where {$_.Name -like "*Path*"}
Get-Process|Select ProcessName,Path|Where {$_.Path -ne $null }
Get-Process | Where starttime -gt (Get-Date '06/06/2016 11:00:00') | select name,
starttime
```

2. Arrêter un process

```
Stop-Process 3512
```

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

3. Verbosité/Erreur

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

4. Arrêter toute une liste de process

```
Get-Process|Select ProcessName|ForEach {
    If ($_.ProcessName -in 'iexplore')
    {
        Stop-Process -processname $_.ProcessName -Verbose -Force
    }
}
```

D. Informations

```
Get-Host : donne notamment les informations de version du PowerShell
Get-Hotfix
Get-HotFix|Where {$_.InstalledOn -lt "03/26/2016"}
Get-Hotfix|Where Description -Like 'Security*'|Select HotFixID,InstallDate,InstalledBy,
FixComments
```

E. Installation de modules

Les modules sont stockés dans C:\Windows\System32\WindowsPowerShell\v1.0\Modules\.

1. Méthodes

```
Get-WindowsFeature|Get-Member -MemberType Property
Get-WindowsFeature|Format-List -Property *
Get-WindowsFeature|Select Name,Installed
Get-WindowsFeature|Where Installed -eq $true|Select Name, DisplayName|Sort Name
Get-WindowsFeature|Where DisplayName -Like '*2.0*'
Install-WindowsFeature -Source C:\Windows\WinSxS -Name NET-Framework-Core
Uninstall-WindowsFeature -Name AD-Domain-Services -Restart -Force

Get-WindowsFeature|Where DisplayName -Like '*hyper*'
Uninstall-WindowsFeature -Name Hyper-V -Restart
Uninstall-WindowsFeature -Name FS-SMB1
```

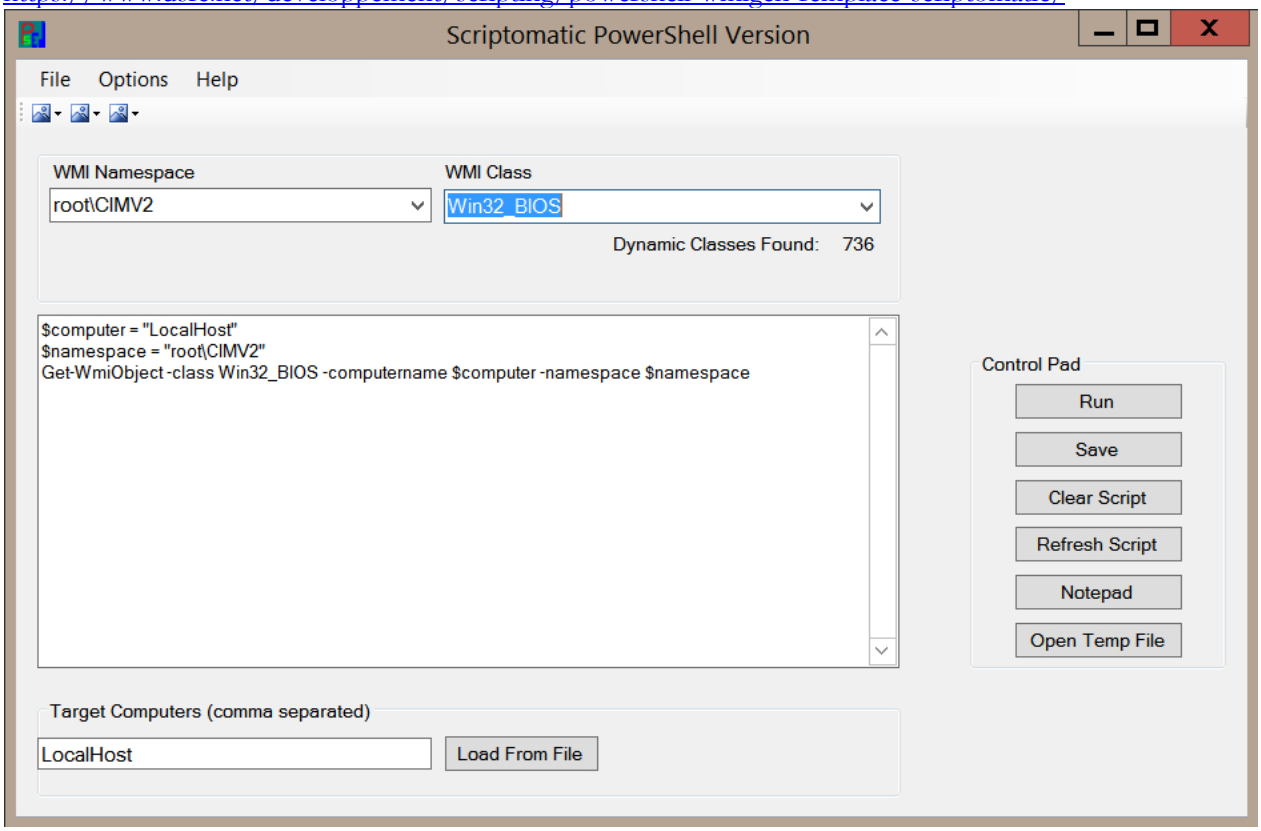
2. Déploiement

```
#(Get-WindowsFeature | Where InstallState -eq 'Installed'|Select-Object Name).Name|Set-Content -
Path 'f:\liste-modules-w2k2012r2.txt'
$fichier=Get-Content -Path 'f:\liste-modules-w2k2012r2.txt'
$liste='DNS','Telnet-Client'
$modules=(Get-WindowsFeature | Where InstallState -eq 'Installed'|Select Name).Name
#ForEach($ligne in $fichier)
ForEach($ligne in $liste)
{
    If(-not ($ligne -in $modules))
    {
        Install-WindowsFeature -Name $ligne
    }
}
```

F. CIM/WMI

1. Scriptomatic pour PowerShell / WMIGen

Il est devenu difficile de le télécharger. Vous pouvez encore le télécharger sur le site du CNET :
http://telecharger.cnet.com/Windows-PowerShell-Scriptomatic/3000-2383_4-75452611.html
WMIGEN remplace Scriptomatic PowerShell.
https://www.dsfc.net/developpement/scripting/powershell-wmigen-remplace-scriptomatic/



2. CIM (Component Information Model)

```
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
```

3. WMI (Windows Management Instrumentation)

```
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 "_*"

4. Exemples

a) Espace disponible

```
Get-WmiObject -List | Where Name -like '*Logical*'
Get-WmiObject Win32_LogicalDisk|Where DriveType -EQ 3|Select DeviceID,FreeSpace
```

```
$data = Get-WmiObject Win32_OperatingSystem
$share = Get-WmiObject Win32_Share
```

b) Pourcentage CPU

```
$cpu = (Get-WmiObject win32_processor | select-object loadpercentage).loadpercentage
```

c) Mémoire dispo

```
$availMem =( Get-WmiObject win32_perfFormattedData_perfos_memory | select-object
availableMbytes).availableMBytes / 1024
Get-WmiObject Win32_PhysicalMemory|Measure Capacity -Sum|Select Sum
```

d) Inventaire logiciel

```
Get-WmiObject Win32_Product | Select Vendor,Caption,Version|Where Caption -like '*Office*'|Sort
Vendor,Caption,Version
Get-WmiObject Win32_Product | Select Vendor,Caption,Version|Where Caption -like '*Office*'|Sort
Vendor,Caption,Version
```

e) Clés USB connectées

```
Get-WmiObject -class Win32_USBHub|Where Caption -Like '*stockage*'
Get-WmiObject -class Win32_USBControllerDevice|Where __PATH -Like '*STOR*'
```

f) Pour obtenir la clé de Windows 8.1

```
powershell "(Get-WmiObject -query ‘select * from
SoftwareLicensingService’).OA3xOriginalProductKey"
```

III. Eléments du langage

A. Les variables et les constantes

1. Les variables

```
[long]$Mem = (Get-WmiObject Win32_ComputerSystem).TotalPhysicalMemory
```

```
$Mbyte =1048576 # Another variable
"Memory Mbyte " + [int]($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
```

```
[float]$CumulTempsCPU=0
$Processes=Get-Process|Select ProcessName,CPU|Sort CPU -Descending
ForEach($Process In $Processes)
{
    $CumulTempsCPU+=$Process.CPU
}
<#
$CumulTempsCPU=(Get-Process|Select CPU|Measure-Object CPU -Sum).Sum
$Processes=Get-Process|Select ProcessName,CPU|Sort CPU -Descending
#>
ForEach($Process In $Processes)
{
    If($Process.CPU -gt 1)
    {
        $Process.ProcessName+' : '+($Process.CPU/$CumulTempsCPU)*100
    }
}
```

2. Les types

Float, int, datetime, string, Boolean, Array, Object, Int32, Int64, Long, Double, Single, **PSCustomObject**

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

3. Transtypage (cast)

```
Clear-Host
[int]$a='11'
$a -is [string]
$a=[string]$a
$a -is [string]
```

4. 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

"La mémoire de la machine est $((Get-WmiObject Win32_ComputerSystem).TotalPhysicalMemory)"


#Here-String
$texte=@'
hgfhgh
gjgjjgj
'@

$texte=@"
Hgfhgh $b
gjgjjgj
"@
```

5. Caractères spéciaux

Ils ne peuvent être utilisés qu'à l'intérieur des guillemets.

`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
`$	Dollar

```
Clear
$computer = "."
$namespace = "root\CIMV2"
$logiciels=Get-WmiObject -class Win32_SoftwareFeature -computername $computer -namespace
$namespace|Select Vendor,ProductName,Version|Sort Vendor,ProductName,Version -Unique
#$logiciels|Select Vendor,ProductName,Version
ForEach($logiciel in $logiciels)
{
    "`"$($logiciel.Vendor)`",`"$($logiciel.ProductName)`",`"$($logiciel.Version)`""
}
$logiciels=$null
```

6. Substitution de variables

```
Clear
$file=Get-ChildItem c:\windows\WindowsUpdate.log
$taille=$file.Length/1024
$path=$file.FullName
"Taille du fichier $path : $taille ko"
'Taille du fichier '+$path+' : '+$taille+' ko'
"Taille du fichier {1} : {0} ko" -f [Math]::Round($taille),$path
```

7. 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
$Env	Tableau des variables d'environnement
$Error	Liste des erreurs de la session
$Event	Evénement traité par Register-ObjectEvent

$EventArgs	Arguments relatifs à Event
$Foreach	Enumerateur d'une boucle ForEach
$Home	Répertoire de base de l'utilisateur
$Host	Informations sur l'hôte
$LastExitCode	**Code de sortie de la dernière commande du système execute**
$PID	Process du script PowerShell
$Profile	Chemin du profil PowerShell
$PSHome	Répertoire d'installation du PowerShell
$PSItem ou $_	Objet courant (ligne du résultat d'un CmdLet par exemple)
$PSScriptRoot	Répertoire du script
$PSVersionTable	Information sur PowerShell
$PWD	Répertoire courant
$ShellID	Identificateur du Shell
$MyInvocation	$MyInvocation.MyCommand.Name
$null	
$false	
$true	
$global	

8. Les constantes

```
Set-Variable Thermometer 32 -option constant
```

9. Les variables globales

```
Set-Variable AllOverPlace 99 -scope global
$global:runners = 8
```

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
$tab=1,2,3,4
$tab=$tab[0..1+3]
$tab=$tab|Where-Object {$_ -ne 3}
$tab=$null #Destruction du tableau
Tableau vide : [string[]]$tab=@()
```

2. Tableau de tableaux

```
[int[]]$tab1=1,2,3,4
[int[]]$tab2=1,2,3,4
$tab=$tab1,$tab2
$tab[0][0]
```

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

Clear
$Jours='Lu','Ma','Me','Je','Ve','Sa','Di'
#$Jours[0]
$n=$Jours.Length
#$n=$Jours.Count

<#
ForEach($Jour in $Jours)
{
    $jour
}
#>
For($i = 0; $i -LT $n; $i=$i+1)
{
    $Jours[$i]
}
```

4. Tableau PowerShell : PSCustomObject

```
Clear
$Custom=@()
$Custom+=[PSCustomObject] @{Indice=1; Valeur='Valeur 1'}
$Custom+=[PSCustomObject] @{Indice=2; Valeur='Valeur 2'}
$Custom|Out-GridView

Clear
$Softs=@()
$Rows=(Get-WMIObject -class 'Win32_Product'|Select Version, Caption)
#$Rows[2]
$i=0
ForEach($Row In $Rows)
{
    $Softs+=[PSCustomObject] @{Caption=$Row.Caption; Version=$Row.Version}
}
$Rows=$null
#$Softs[2]
$Softs|Out-GridView
Remove-Item -Path 'd:\liste-logiciels.txt'
$Softs|Export-Csv -Path 'd:\liste-logiciels.txt' -NoTypeInformation
$Softs=$null
```

5. 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
```

6. 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 $_}
```

7. Autres méthodes

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

8. Portée

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

C. Nombre aléatoire

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

D. Opérateurs

1. Modulo

%

2. Concaténation

+ Le signe plus est l'opérateur de concaténation en PowerShell.

3. 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

4. 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

Clear
$cmd=&{ipconfig /all}
<#
$cmd=@'
Test
Chaîne AE345-4678A-7689D
Gazou
'@
#>
#$cmd
ForEach($row in $cmd)
{
    #if($row -match '([0-9A-F]{2}-[0-9A-F]{2}-[0-9A-F]{2}-[0-9A-F]{2}-[0-9A-F]{2}-[0-9A-
F]{2})')
    if($row -match '(([0-9A-F]{2}-){5}[0-9A-F]{2})')
    #if($row -match '([0-9A-F\-]{17})')
    {
        $Matches[1]
    }
}
```

5. Logiques

-and Et
-or Ou
-xor Ou exclusif

6. Plages

1..99 Un intervalle de 1 à 99 !

7. Appartenance

```
'DSFC' -in 'DSFC','Szalkowski'
'DSFC' -notin 'DSFC','Szalkowski'
```

Contains, c'est l'inverse :

```
'DSFC','Szalkowski' -contains 'DSFC'
```

8. Opérateurs binaires

```
-band
-bor
-bnot
-bxor
```

9. Affectation

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

10. Cast / Transtyper

```
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
```

11. Forcer la définition de variables

```
Set-PSDebug -Strict
```

E. 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 "

Clear
$services=Get-Service | Select Name,RequiredServices
ForEach($service in $services)
{
    $RequiredServices=$service.RequiredServices
    $tmp=$service.Name+':'
    Foreach($RequiredService in $RequiredServices)
    {
        $tmp+=$RequiredService.Name+','
    }
    $tmp
}
```

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'
}
```

F. 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
```

3. Trap

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

4. Try…Catch

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

5. Débogage

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

G. Pipelining avancé

1. Comptage

```
Get-Service | Group-Object Status|Select Name,Count
Get-ChildItem c:\windows | Group-Object extension
```

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

2. Statistiques

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

Clear-Host
$res=@()
$rows=(Get-Process |Select ProcessName,PrivateMemorySize)
$Processes=($rows|Select ProcessName -Unique)
ForEach($Process in $Processes)
{
    $mem=($Rows |Where ProcessName -eq $Process.ProcessName`
    |Measure PrivateMemorySize -sum).Sum/1048576

    $res+=[PSCustomObject] @{Process=$Process.ProcessName;Ram=$mem}
}
$rows=$null
$Processes=$null
$res|Sort Ram -Descending|Out-GridView
$res=$null
Clear
$res=@()
$Processes=(Get-Process|Select ProcessName,PrivateMemorySize)
$n=$Processes.Length
$i=0
while($i -lt $n)
{
    $p=($Processes[$i]).ProcessName
    $ram=0
    do
    {
        $ram+=($Processes[$i]).PrivateMemorySize
        $i++
    }
    while($p -eq ($Processes[$i]).ProcessName)
    $res+=[PSCustomObject] @{Process=$p;Ram=$ram/1048576}
}
$Processes=$null
$res|Out-GRidView
$res=$null
```

3. Sélection

```
Get-Process|Select-Object ProcessName -first 5
Get-Process|Sort CPU -Descending|Select-Object ProcessName -first 10
```

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 Name
Get-Service | Format-List * #Liste toutes les propriétés
```

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
#Sortie graphique
Get-Process|Select *|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

#Script suggéré par Rodolphe
Filter FiltreCompteur
{
    Begin
    {
        $i=0
    }
    Process
    {
        If($_.Status -eq 'Running')
        {
            $_
            $i++
        }
    }
    End
    {
        "`nTotal des services s'exécutant : $i"
    }
}
Get-Service|FiltreCompteur|Format-Table -Property Name
```

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
Get-Process|Select Name -Unique
```

9. Propriétés

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

10. Impressions

```
Get-Process | Out-Printer
Get-Process | Out-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
```

Dans les messages, les couleurs utilisables sont :

```
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
```

H. Fonctions

1. Sans retour

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

I. Gestion des modules

1. Emplacement des modules

Ils sont déterminés par la variable d'environnement $env:PSModulePath.

```
%windir%\System32\WindowsPowerShell\v1.0\Modules
%UserProfile%\Documents\WindowsPowerShell\Modules
```

2. Télécharger des modules complémentaires

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

3. Les modules liés à l'administration

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

4. Commandes d'un module

```
Get-command –module DnsServer
```

5. Charger automatiquement les modules

```
$PSModuleAutoloadingPreference='All' (None,ModuleQualified)
```

6. Décharger un module

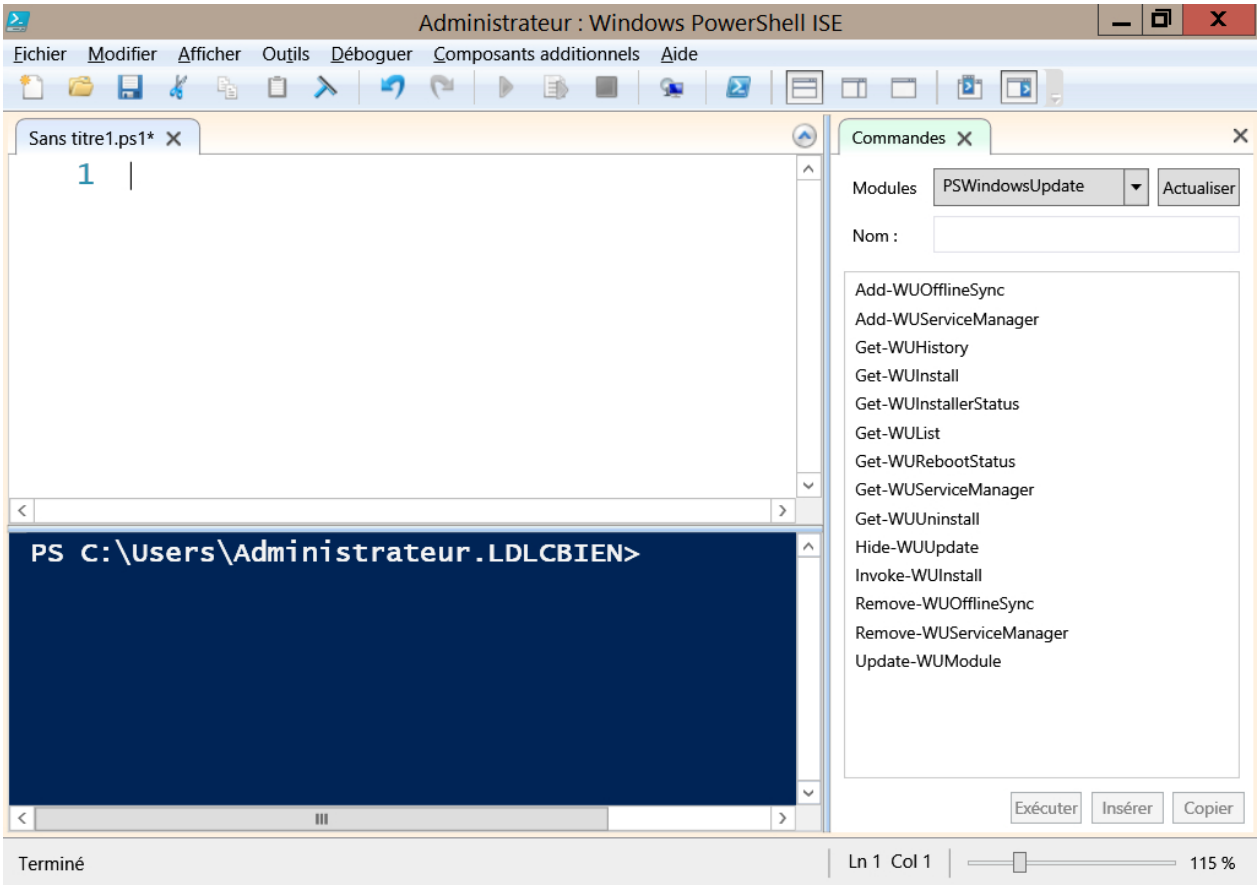
```
Remove-Module Dnsserver
```

7. Créer un module

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

8. Utilisation du module PSWindowsUpdate

Vous devez tout d'abord le télécharger à partir de la gallerie Microsoft dans le répertoire C:\Windows\system32\WindowsPowerShell\v1.0\Modules, puis le dézipper. Après actualisation dans PowerShell ISE, son nom apparâit dans la liste des modules.



9. Exemple : devices

a) Le fichier devices.psd1

```
@{
#GUID = 'bae93d8e-782c-4a23-b87f-8699bfc17ee0'
Author = 'DSFC'
CompanyName = 'DSFC'
Copyright = 'DSFC'
ModuleVersion = '1.0'
PowerShellVersion='3.0'
CLRVersion='4.0'
RootModule='devices.psm1'
FunctionsToExport='Get-DriveFreeSpace', 'Get-Ram'
}
```

b) Le fichier devices.psm1

```
<#
    .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
}
<#
    .Synopsis
    Indique la conso mémoire d'un processus.

    .Description
    La fonction Get-Ram indique la mémoire consommée par un processus.

    .Parameter Process
    Entrez le nom du processus comme Firefox.

    .Example
    Get-Ram 'Firefox'

    .Example
    Get-Ram -process 'Firefox'

    .Link
    Get-Ram
#>
Function Get-Ram
{
    Param([string]$process)
    Return (Get-Process|Where Name -EQ $process |Measure WorkingSet -Sum).Sum
}
```

c) Utilisation du module

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

IV. 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
```

Exemple : `(Get-Date 1/1/2016).AddDays(5)`

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

Le dollar est facultatif.

```
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)
```

Extraire les ouvertures de session du journal d'événements :

```
Get-EventLog -LogName Security -After (Get-Date).AddDays(-1)|Select Message,TimeWritten|Where
{$_.Message -match "L'ouverture de session"}|Out-GridView
```

```
Get-EventLog -LogName Application -After (Get-Date).AddDays(-3) -Before (Get-Date).AddDays(-2)
|Where EntryType -Match 'Error'
```

E. Filtre sur dates

```
Get-EventLog -LogName Application -EntryType Error |Select TimeWritten,Message|Where
{$_.TimeWritten -GE (Get-Date 18/7/2017) -AND $_.TimeWritten -LT (Get-Date 19/7/2017)}
```

F. Création de fichier avec la date du jour

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

V. Gestion de fichiers

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

A. Système

1. Se déplacer sur le système de fichiers

```
Set-Location -Path 'd:/download'
Set-Location d:\
```

2. 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
```

3. Création de fichiers et de répertoires : 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
```

4. 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
```

5. Renommer les fichiers

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

6. Recherche de fichiers

```
Get-ChildItem -Path c:\ -recurse| Select Name,FullName|Where {$_.Name -match
'WindowsUpdate\.log'}
```

7. 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*

Clear
If($env:TEMP -ne $null)
{
    Remove-Item  "$($env:TEMP)\*.*" -Force -Recurse
}
If($env:TMP -ne $null)
{
    Remove-Item  "$($env:TMP)\*.*" -Force -Recurse
}
```

8. Copie récursive

```
Clear
$src='c:\windows'
$dst='d:/temp'
If (-not $(Test-Path -Path $dst))
{
    New-Item -Path $dst -itemType directory
}
Set-Location -Path $dst
Get-ChildItem -Path "$src\*.log" -Recurse|Copy-Item -Destination $dst -Force
Get-ChildItem -Path $dst
```

9. Suppression récursive

```
Clear
$exts='bak','tmp'
ForEach($ext IN $exts)
{
    Get-ChildItem -Path "c:\*.$ext" -Recurse -ErrorAction SilentlyContinue|Remove-Item -Force
}
```

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
If( -not $(Test-Path C:\Users\Administrateur\Desktop\f.txt))
{
   New-Item C:\Users\Administrateur\Desktop\f.txt -type file
}
```

D. Lire un répertoire

1. Commandes

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

2. Attributs (IO.FileAttributes)

- ReadOnly
- Hidden
- System
- Directory
- Archive
- Device
- Normal

• Temporary
• SparseFile
• ReparsePoint
• Compressed
• Offline
• NotContentIndexed
• Encrypted

E. La sécurité

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

&{icacls C:\Users\Administrateur\Desktop\d.txt /grant denis:F}
$acls=Get-Acl -Path 'C:\Users\Administrateur\Desktop\d.txt'
Set-Acl -Path 'C:\Users\Administrateur\Desktop\f.txt' -AclObject $acls
Get-Acl 'C:\Users\Administrateur\Desktop\f.txt'
```

F. Ajout à un fichier

La méthode Add-Content fait un retour à la ligne automatique (\r\n).

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

G. Recherche dans le contenu d'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
Select-String -Path 'c:\windows\WindowsUpdate.log' –Pattern '(FATAL|WARNING)'
```

H. Visualiser le contenu d'un fichier

```
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
Get-Content 'c:\windows\ntbtlog.txt'|Select-String –Pattern 'BOOTLOG_NOT_LOADED'|Sort|Get-Unique
Select-String –Path 'c:\windows\ntbtlog.txt' –Pattern 'BOOTLOG_NOT_LOADED'
```

I. Les redirections

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

J.Création d'un fichier

La différence entre Out-File et Set-Content est que le premier ne sait créer que des fichiers texte. Out-File récupère le résultat d'une commande au format texte.

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

K. Effacer le contenu d'un fichier

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

L. 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
```

M. Conversion en JSON

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

N. Compter les lignes d'un fichier

```
Get-Content c:\config.sys | Measure-Object
```

O. Lire les 5 dernières lignes d'un fichier

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

P. Filtrer des lignes

```
Clear
Get-Content -Path E:\unbound.log|Where {$_ -match 'clients1\.google\.com'} <# |Measure-Object #>
```

Q. 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"}
```

R. Les fichiers XML

```
Get-ChildItem d:\scripts | Export-Clixml d:\scripts\files.xml
$A = Import-Clixml d:\scripts\files.xml
$A | Sort-Object length
Get-Process | Export-Clixml d:\scripts\test.xml
```

S. 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"
```

```
Get-Process | Export-Csv -Path $file -Delimiter ';' -Encoding UTF8 -NoTypeInformation
 #TYPE System.Diagnostics.Process
Get-Process | Export-Csv d:\scripts\test.txt -notype
Get-Process | Export-Csv d:\scripts\test.txt –force
```

T. Sauvegarde d'un fichier

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

U. 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
```

V. Interactif

```
Get-Service|Out-GridView
```

W. Export / Import CSV Tableaux et Tableaux associatifs

Pour importer un fichier CSV, vous disposez de l'instruction Import-CSV. A noter qu'à partir de la version 4 du PowerShell, les lignes vides sont ignorées.

X. Obtenir le Hash d'un fichier

L'algorithme utilisé peut être le MD5, le SHA1 ou le SHA256

```
Get-FileHash -Algorithm <Nom-algorithme> -Path <Chemin-vers-fichier>
```

VI. 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" -PropertyType String
$path='HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer'
$val = Get-ItemProperty -Path $path -Name 'AlwaysUnloadDll'
If($val -eq $null)
{
    New-ItemProperty -path $path -name 'AlwaysUnloadDll' -value 1 -PropertyType Dword
}
Else
{
    Set-Itemproperty -Path $path -Name "AlwaysUnloadDll" -value 1
}
```

D. Suppression de clé

```
Remove-Item
```

E. Lecture / Ecriture

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

F. Exemples

```
$ErrorActionPreference='SilentlyContinue'
$path='HKLM:\SYSTEM\CurrentControlSet\Services\Tcpip6\'
Set-Location -Path $path
If(-not $(Test-Path -Path '.\Parameters'))
{
    New-Item -Path . -Name Parameters
}
$path+='Parameters\'
Set-Location -Path $path
Clear
Remove-ItemProperty -path . -name DisabledComponents
New-ItemProperty -path . -name DisabledComponents -value 0 -PropertyType DWord

Clear
$path='HKLM:SYSTEM\CurrentControlSet\Services\USBSTOR'
$usb=Get-ItemProperty -Path $path -name 'Start'
```

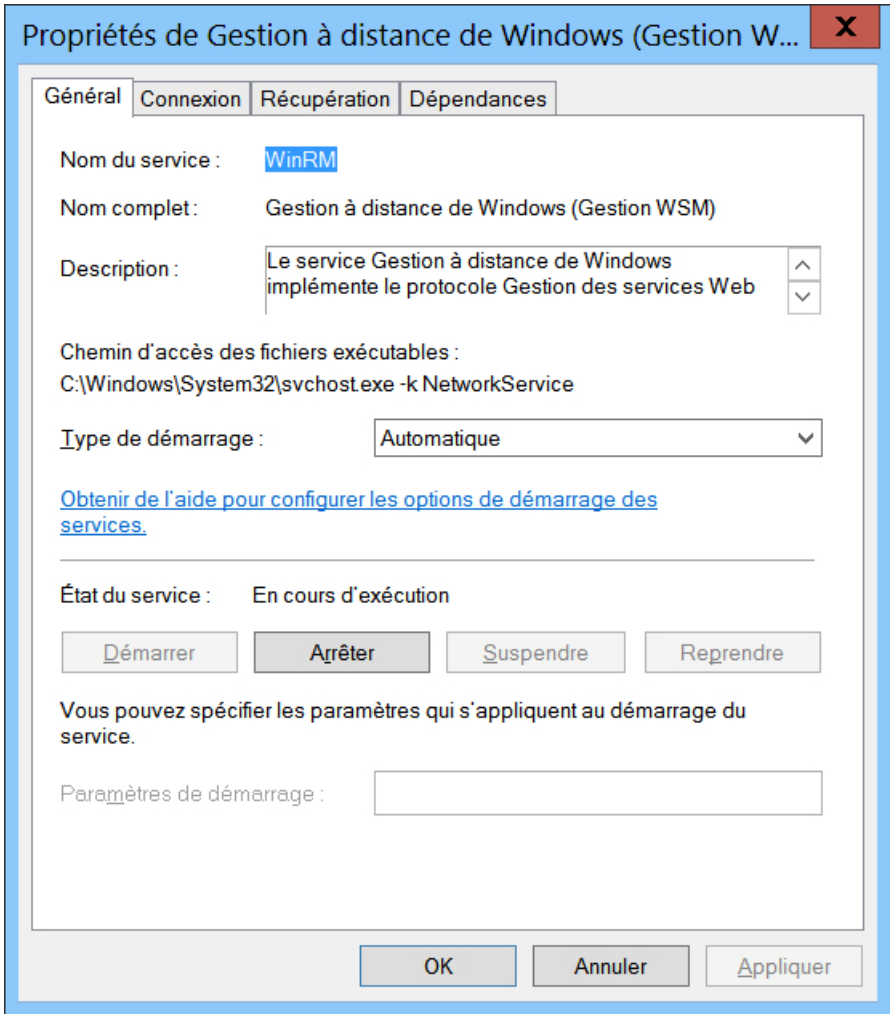
```
If($usb.Start -ne 4)
{
    Set-ItemProperty -Path $path -name 'Start' -value 4
}

Clear
$path='HKLM:SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer'
Set-Location -Path $path
#Remove-ItemProperty -path . -name AlwaysUnloadDll
New-ItemProperty -path . -name AlwaysUnloadDll -value 1 -PropertyType DWord -Force
```

VII. Exécution distante

A. Présentation

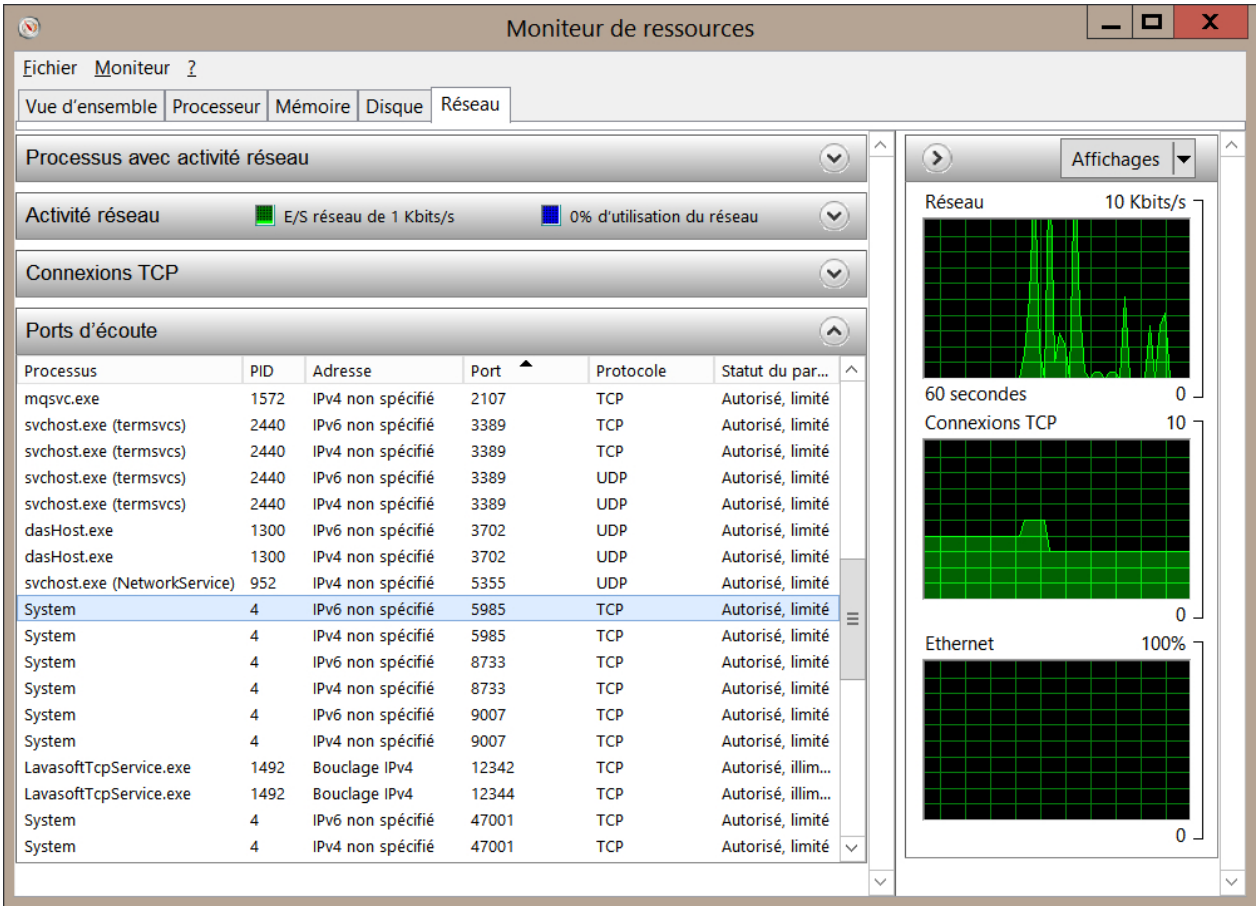
Powershell utilise le RPC Remote Procedure Call. Il s'appuie sur le service WinRM (Gestion à distance de Windows). Il utilise les ports 5985 et 5986, en mode sécurisé.



Au niveau du pare-feu, vérifiez que les règles liées à la gestion distante soient activées.
Pour vérifier que le service s'exécute, tapez : netstat –ano|find "5985" :

```
C:\Users\Administrateur>netstat -ano|find "5985"
  TCP    0.0.0.0:5985           0.0.0.0:0              LISTENING       4
  TCP    [::]:5985              [::]:0                 LISTENING       4
```

Vous pouvez aussi utiliser le moniteur de performances.



1. Configuration

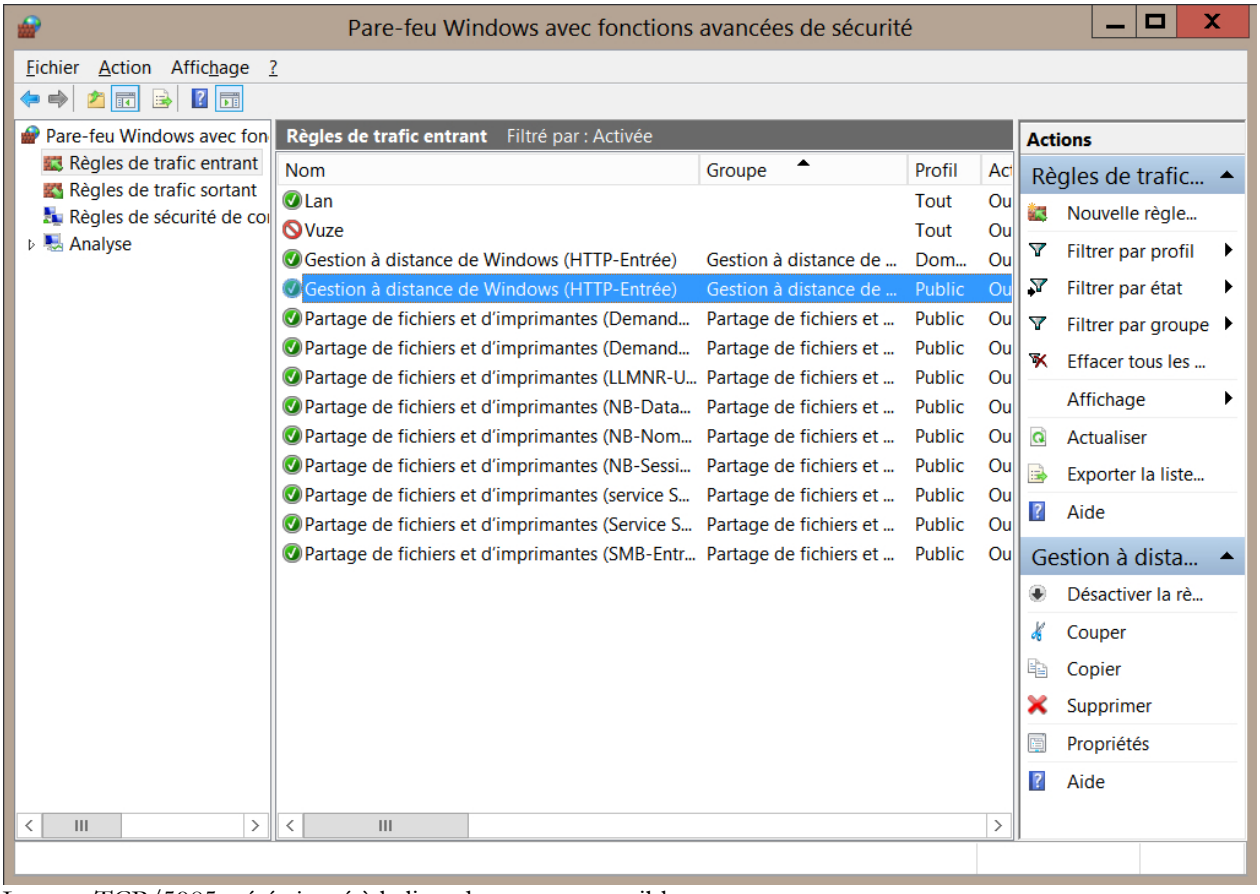
Pour configurer le service, tapez sous Powershell : Enable-PSRemoting. Vous disposez aussi de la commande winrm quickconfig.
Pour vérifier la configuration : winrm get winrm/config

2. Sécurité

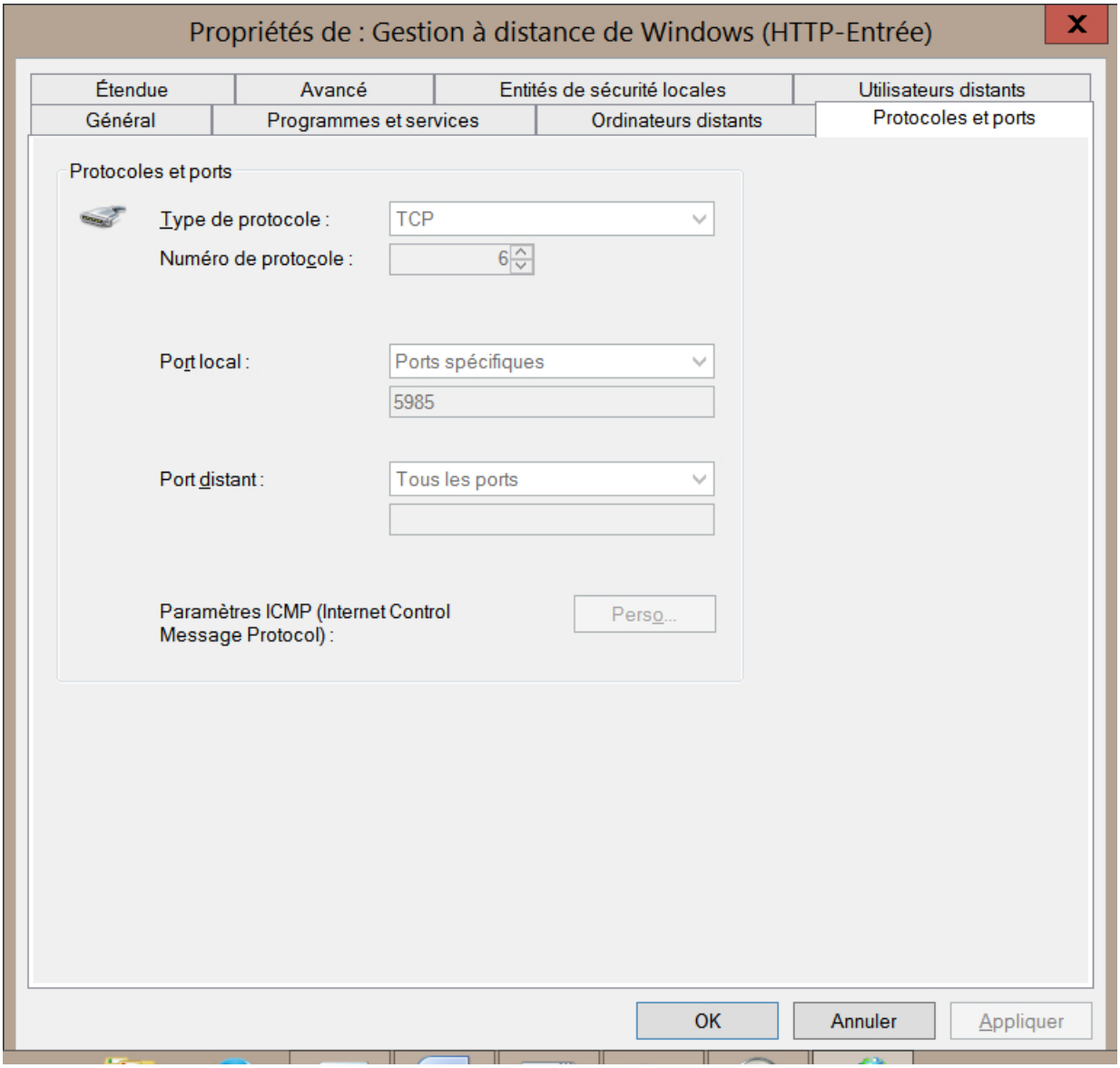
```
Enable-PSRemoting -Force
```

3. Règle de pare-feu

L'exécution de winrm ajoute une règle au pare-feu :

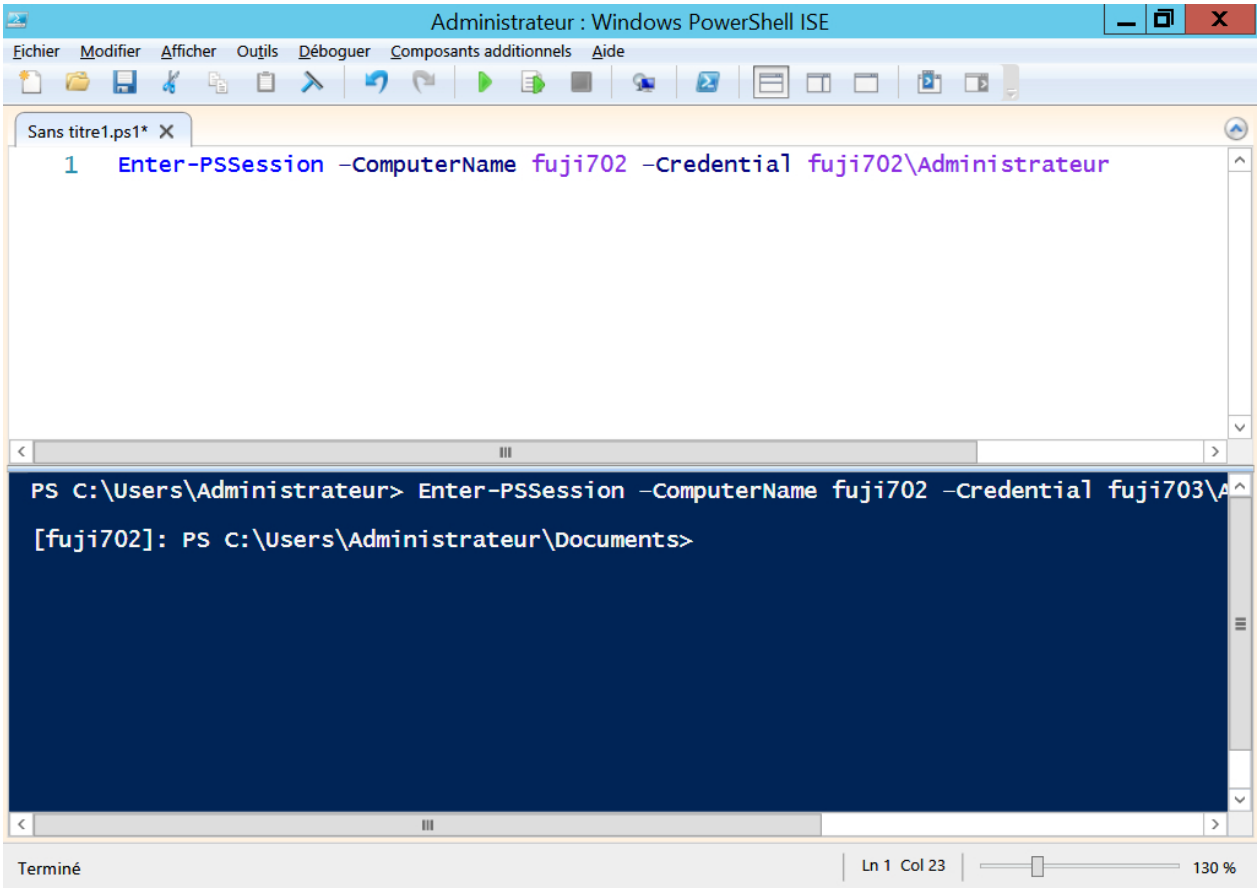


Le port TCP/5985 a été ajouté à la liste des ports accessibles.



B. Ouverture de session distante

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



C. Authentification

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

D. Machines de confiance (Poste à poste)

Sur le client :

```
Set-Item WSMan:\localhost\client\trustedhosts -value ACERARIEN -force -Concatenate
Set-Item WSMan:\localhost\Client\TrustedHosts *
Get-Item WSMan:\localhost\client\trustedhosts
```

Pour vérifier: winrm get winrm/config

```
winrm set winrm/config/Client @{AllowUnencrypted = "true"}
```

Au niveau du registre, passez le paramètre

HKLM\SOFTWARE\Microsoft\windows\CurrentVersion\Policies\System\LocalAccountTokenFilterPolicy à 1 (DWord)

En Powershell :

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

E. Droits

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

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

F. 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. Sortir de la session

```
Exit-PSSession
```

4. Exécution distante

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

5. Rappel de la session

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

G. Liste des commandes possibles

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

H. Détruire les sessions distantes sur la machine

```
Stop-Process -processname wsmprovhost -Force
Get-PSSession
Remove-PSSession -Id 1,3
```

I. Nombre de connexions simultanées

```
winrm get winrm/config -> MaxConcurrentUsers
winrm set winrm/config/winrs @{MaxConcurrentUsers="20"}
```

J.Exemples

1. Invoke-Command

```
Get-PSSession|Remove-PSSession
$motdepasse = ConvertTo-SecureString "password" -AsPlainText -Force
$authentification = New-Object
System.Management.Automation.PSCredential("MF230\Administrateur",$motdepasse)
$session=New-PSSession –ComputerName  '192.168.43.126' -Credential $authentification
$path=Invoke-Command -ScriptBlock {$env:computername} -Session $session
$cmd=Invoke-Command -ScriptBlock {&ipconfig} -Session $session
clear
$path
$cmd
 Get-PSSession|Remove-PSSession
```

2. Get-Process

```
$motdepasse = ConvertTo-SecureString "password" -AsPlainText -Force
$authentification = New-Object System.Management.Automation.PSCredential `
("MF230\Administrateur",$motdepasse)
```

```
Enter-PSSession –ComputerName MF230 -Credential $authentification
Get-Process -ComputerName MF230
```

3. Inventaire

```
Get-PSSession|Remove-PSSession
$motdepasse = ConvertTo-SecureString "admin" -AsPlainText -Force
$machines='fuji702','fuji703','uc-neuro2-pc','tour-na1-pc'
$file='d:\processes.txt'
If($(Test-Path -Path $file))
{
    Clear-Content -Path $file
}
foreach($machine in $machines)
{

    $authentification = New-Object
System.Management.Automation.PSCredential("$machine\Administrateur",$motdepasse)
    $session=New-PSSession –ComputerName $machine -Credential $authentification
    $cmd=Invoke-Command -Session $session -ScriptBlock {
        Get-Process
    }
    $cmd|Select ProcessName,PSComputerName -Unique|Export-Csv -Path $file -Append -
NoTypeInformation
    Get-PSSession|Remove-PSSession
}
```

VIII. Modules Windows 8 et 2012

A. NetAdapter

1. Importer le module NetAdapter

```
Import-Module NetAdapter
```

2. Profil

```
Get-NetConnectionProfile
```

3. Lister les périphériques réseaux

```
Get-NetAdapter
```

4. Lister les interfaces IP

```
Get-NetIPInterface
Get-NetIPConfiguration
```

5. Elements attachés à la carte réseau

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

6. Désactiver IPv6

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

7. Définir une adresse Ip

```
New-NetIPAddress –InterfaceIndex 27 –IPAddress 192.168.1.100 –PrefixLength 24 –DefaultGateway
192.168.1.1
```

8. Passer en DHCP

```
Set-NetIPInterface -InterfaceIndex 27  -Dhcp {Enabled/Disabled}
```

9. Supprimer une Ip

```
Remove-NetIPAddress –InterfaceIndex 27 –IPAddress 192.168.1.100 –PrefixLength 24 –DefaultGateway
192.168.1.1
```

10. Changer le DNS

```
Set-DnsClientServerAddress -InterfaceIndex 27 -ServerAddresses 192.168.1.1
Get-DnsClientServerAddress -InterfaceIndex 27
```

B. NetConnection

```
Clear
$if=(Get-NetConnectionProfile).InterfaceIndex
Get-DnsClientServerAddress -InterfaceIndex $if
```

C. 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
```

D. Impression

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

E. ODBC

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

F. DNS

```
Resolve-DnsName –Name yahoo.fr | Format-List
Resolve-DnsName –Name gmail.com -Type MX| Where IPAddress -ne $null |Select IPAddress|Out-
GridView
Get-DnsClientCache| Select-Object -Property Name
Get-DNSClientServerAddress|Where-Object ServerAddresses
```

G. Disque

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

H. Drivers

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

I. Applications

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

Pour désinstaller une application, tapez :

```
Remove-AppxPackage NomDuPackageADesinstaller
```

J.Le BPA Best Pratice Analyzer (Windows Server 2012)

```
Get-Command -Module BestPractices
Get-BpaModel|Invoke-BpaModel
Get-BpaResult Microsoft/windows/DNSServer
```

K. Panneau de configuration

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

L. Renommer un ordinateur

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

M. Windows Core

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

N. Liste de tous les composants installés

```
Get-WindowsFeature
```

IX. 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
```

b) *Membre d'un groupe*

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

c) *Ajout à un groupe*

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

d) *Supprimer un membre d'un groupe*

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

e) *Lister les utilisateurs*

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

f) *Créer un groupe*

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

g) *Renommer un groupe*

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

2. Gestion des utilisateurs

a) 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()
```

b) Modifier un compte local

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

c) Lister les propriétés d'un utilisateur

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

B. Installation sur Windows 7 du module ActiveDirectory

Il faut au préalable installer les outils d'administration de serveur distant pour Windows 7 SP1.
https://www.microsoft.com/fr-FR/download/details.aspx?id=7887

C. 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'
```

b) Requêtes

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

```
#Interrrogation sur un autre domaine (suppose relation d'approbation)
Get-ADObject -LDAPFilter '(&(objectCategory=person)(objectClass=user))' -SearchBase
'DC=dsfc,DC=edu'
Get-ChildItem -Path 'AD:\CN=Users,DC=madeinchina,DC=local' -Recurse|Where ObjectClass -eq 'user'
Get-ChildItem -Path 'AD:\DC=madeinchina,DC=local' -Recurse|Where ObjectClass -eq 'computer'
```

c) Filtres

Aide sur les filtres : `Get-Help about_ActiveDirectory_Filter`

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

Pour la liste des comptes désactivés :

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

d) 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*'}}
```

e) 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
```

f) 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
```

g) 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
```

d) Effacer un utilisateur

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

e) lire les attributs

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

f) Modifier des attributs

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

g) effacer un attribut

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

h) La gestion des utilisateurs

```
Get-ADUser UserName : Liste les informations relatives à un nouvel utilisateur
Get-ADUser -Filter {Name -like "*SearchVariables*"} : Filtrage des informations d'un utilisateur
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 : création d'un Utilisateur
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 : import d'un fichier CSV
Remove-ADUser *UserName* : efface un Utilisateur
Set-ADUser *ADUser -Variable* : modifie les propriéts d'un utilisateur
Set-ADUser ben.jones -Office Brisbane

5. Les groupes

a) Commandes relatives aux groups

```
Get-Command –Module ActiveDirectory –Name *group*
```

b) 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=dutout,DC=net'
```

c) Création de groupes

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

d) Membres d'un groupe

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

e) Ajout à un groupe

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

f) Supprimer les membres d'un groupe

Pour ces deux commandes, vous pouvez utilizer le paramètre –Confirm:$false

```
Remove-ADGroupMember
Remove-ADPrincipalGroupMemberShip
```

g) Suppression d'un groupe

```
Remove-ADGroup
```

h) Les groupes

Get-ADGroup *GroupName* Liste les informations d'un groupe

Get-ADGroup -Filter {Name -like "**SearchVariables**"} : Applique un filter à un groupe
Exemple : Get-ADGroup -Filter {Name -like "*mins*"}

New-ADGroup -name *GroupName* -GroupScope *Global|Universa*l -Description *"Description"* -DisplayName *DisplayName* -SamAccountName *AccountName* : création d'un groupe
New-ADGroup -name TestGroup -GroupScope Global -Description "New Group Test" -DisplayName TestGroup -SamAccountName TestGroup

Remove-ADGroup *GroupName* : efface un groupe

Set-ADGroup *GroupName -Variable* : modifie les propriétés d'un groupe
Set-ADGroup TestGroup -Description "Demo Group"

D. Le module NTFSSecurity

https://github.com/raandree/NTFSSecurity/releases

```
Install-Module NTFSSecurity
Get-Command -Module NTFSSecurity
Get-NTFSAccess –Path "C:\Partage"
Get-ChildItem -Path "C:\Partage\" -Recurse | Get-NTFSAccess
Add-NTFSAccess –Path "C:\Partage\" –Account "denis@dsfc.local" –AccessRights Modify
Remove-NTFSAccess –Path "C:\Partage"  –Account " denis@dsfc.local " –AccessRights Modify
Get-NTFSEffectiveAccess C:\Partage
```

E. 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
```

X. PowerShell sous Windows Server

A. Source

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

B. La listes des cmdlets

Cmdlet	Description
Add-ADComputerServiceAccount	Ajout d'un compte de service.
Add-ADDomainControllerPasswordReplicationPolicy	Ajoute à la liste des objets autorisés et interdits du contrôleur RODC
Add-ADFineGrainedPasswordPolicySubject	Applique un stratégie de mot de passe
Add-ADGroupMember	Ajoute un membre à un groupe
Add-ADPrincipalGroupMembership	Ajoute un member à plusieurs groupes
Clear-ADAccountExpiration	Efface la date d'expiration d'un compte AD
Disable-ADAccount	Désactive un compte
Disable-ADOptionalFeature	Désactive une configuration optionnelle.
Enable-ADAccount	Active un compte AD
Enable-ADOptionalFeature	Active une configuration optionnelle
Get-ADAccountAuthorizationGroup	Liste tous les groupes d'un utilisateur.
Get-ADAccountResultantPasswordReplicationPolicy	Info sur la stratégie de replication de mot de passe
Get-ADComputer	Liste les ordinateurs de l'AD.
Get-ADComputerServiceAccount	Liste les comptes de service associés à un ordinateur
Get-ADDefaultDomainPasswordPolicy	Donne la stratégie de mot de passé d'un AD
Get-ADDomain	Obtient le domaine associé à l'AD
Get-ADDomainController	Obtient la liste des contrôleurs
Get-ADDomainControllerPasswordReplicationPolicy	Liste les membres de la stratégie de replication de mot de passé avec un RODC
Get-ADDomainControllerPasswordReplicationPolicyUsage	Liste la stratégie de mot de passé associée au RODC.
Get-ADFineGrainedPasswordPolicy	Liste les strategies de mots de passé renforcées
Get-ADFineGrainedPasswordPolicySubject	Permet d'obtenir la liste des utilisateurs et des groups sur lesquels s'applique la stratégie renforcée
Get-ADForest	Liste une forêt
Get-ADGroup	Obtient la liste des groups de l'AD
Get-ADGroupMember	Obtient les membres d'un groupe
Get-ADObject	Obtient les objets de l'AD
Get-ADOptionalFeature	Obtient les configurations optionnelles de l'Ad
Get-ADOrganizationalUnit	Liste plusieurs UO / OU de l'AD
Get-ADPrincipalGroupMembership	Liste des membres d'un groupe
Get-ADRootDSE	Nom de la racine de l'AD

Get-ADServiceAccount	Liste les comptes de service de l'AD
Get-ADUser	Liste les utilisateurs de l'AD
Get-ADUserResultantPasswordPolicy	Permet de connaître la stratégie de mot de passé associée à un utilisateur
Install-ADServiceAccount	Ajoute un compte de service
Move-ADDirectoryServer	Déplace un serveur dans un autre site
Move-ADDirectoryServerOperationMasterRole	change le FSMO
Move-ADObject	Déplace une objet dans un autre container.
New-ADComputer	Ajoute un ordinateur dans l'AD
New-ADFineGrainedPasswordPolicy	Ajoute un stratégie de mot de passe
New-ADGroup	Ajoute un groupe
New-ADObject	Ajoute un objet à l'AD
New-ADOrganizationalUnit	Crée une nouvelle UO / OU
New-ADServiceAccount	Crée un nouveau compte de service
New-ADUser	Crée un nouvel utilisateur
Remove-ADComputer	Enlève un ordinateur de l'AD
Remove-ADComputerServiceAccount	Enlève un compte de service de l'ordinateur
Remove-ADDomainControllerPasswordReplicationPolicy	Enlève un Utilisateur au niveau de la stratégie de replication avec le RODC
Remove-ADFineGrainedPasswordPolicy	Supprimer une stratégie de mots de passé renforcée
Remove-ADFineGrainedPasswordPolicySubject	Enlève des utilisateurs d'une stratégie de mot de passé renforcée
Remove-ADGroup	Enlève un groupe de l'Ad
Remove-ADGroupMember	Enlève des elements d'un groupe
Remove-ADObject	Efface un objet de l'AD
Remove-ADOrganizationalUnit	Efface une UO / OU
Remove-ADPrincipalGroupMembership	Enlève un member d'un groupe
Remove-ADServiceAccount	Efface un compte de service
Remove-ADUser	Retire un utilisateur
Rename-ADObject	Renomme un objet de l'AD
Reset-ADServiceAccountPassword	Resets the service account password for a computer.
Restore-ADObject	Restores an Active Directory object.
Search-ADAccount	Recherche un objet dans l'AD
Set-ADAccountControl	Modifie l'UAC d'un utilisateur
Set-ADAccountExpiration	Fixe la date d'expiration d'un compte de l'AD
Set-ADAccountPassword	Modifie le mot de passé d'un utilisateur de l'AD
Set-ADComputer	Change les proprieties d'un ordinateur
Set-ADDefaultDomainPasswordPolicy	Modifie la stratégie de mot de passe
Set-ADDomain	Modifie le domaine d'un Ad
Set-ADDomainMode	Change le niveau fonctionnel d'un domaine

Set-ADFineGrainedPasswordPolicy	Rend plus fine la stratégie de mot de passe.
Set-ADForest	Modifie la forêt d'un AD
Set-ADForestMode	Détermine la forêt pour un AD
Set-ADGroup	Modifie un groupe
Set-ADObject	Modifie un objet de l'AD
Set-ADOrganizationalUnit	Modifie une UO /OU
Set-ADServiceAccount	Modifie un compte de service.
Set-ADUser	Modifie un utilisateur
Uninstall-ADServiceAccount	Désinstalle un compte de service
Unlock-ADAccount	Dévérouille un compte AD

XI. Quelques exemples

A. Liste des fichiers exécutés sur la machine

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

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

B. Liste des services à partir du registre

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

C. Utilisation des composants WSH Windows Scripting Host

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

1. Wscript.Shell

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

2. Wscript.Network

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

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

3. Partage d'imprimante

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

4. Scripting.FileSystemObject

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

D. MySQL : lecture de tables

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

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

E. Les compteurs

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

F. MySQL : inventaire

1. La table

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

2. Le script

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

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

XII. Quelques sites

PowerShell 5.0 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 4.5.

A. Sites en français

- Windows PowerShell (**site officiel**) : guide
  https://technet.microsoft.com/fr-fr/library/bb978526.aspx
- Centre de scripts Windows PowerShell (**site officiel**) : téléchargements, scripts, mémento
  https://msdn.microsoft.com/en-us/powershell
- Galerie de scripts PowerShell (**site officiel**) : téléchargements, scripts
  https://gallery.technet.microsoft.com/scriptcenter/site/search?f[0].Value=PowerShell
- Denis Szalkowski Formateur Consultant : tutos, supports
  http://www.dsfc.net/powershell/
- IT-Connect : tutos
  http://www.it-connect.fr/tag/powershell/
- Laurent Dardenne : liens, tutoriaux
  http://laurent-dardenne.developpez.com/articles/Windows/PowerShell/Ressources/
- PowerShell-Scripting.com : articles, tutoriaux, scripts, mémento
  http://powershell-scripting.com/
- via PowerShell : tutoriaux, liens
  http://www.via-powershell.fr/
- SysKB : scripts
  http://syskb.com/tag/powershell/

B. Sites en anglais

- PowerGUI : tutos, wiki, etc
  http://en.community.dell.com/techcenter/powergui
- PowerShell.com : scripts, tutoriaux
  http://powershell.com/cs/media/default.aspx
- Sapien Technologies : téléchargements, scripts (inscription obligatoire)
  https://www.sapien.com/auth/other/downloads
- Precision Computing : scripts
  http://www.leeholmes.com/blog/?s=powershell
- CodePlex (modules PowerShell Open Source) : téléchargements
  http://www.codeplex.com/site/search?query=powershell

C. Téléchargements

- Microsoft Framework .Net 4.5 **(site officiel)**
  https://www.microsoft.com/fr-FR/download/details.aspx?id=42642

- Windows Management Framework 4.0 **(site officiel)**
  https://www.microsoft.com/fr-fr/download/details.aspx?id=40855

- PowerShellPack **(site officiel)**
  http://archive.msdn.microsoft.com/PowerShellPack/Release/ProjectReleases.aspx?ReleaseId=3341

- PowerShell Scriptomatic (en)
  https://technet.microsoft.com/en-us/library/ff730935.aspx

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

A. Les CmdLets

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

Get-Member
Get-Module
Get-NfsMappedIdentity
Get-NfsNetgroup
Get-PfxCertificate
Get-PfxData
Get-Process
Get-PSBreakpoint
Get-PSCallStack
Get-PSDrive
Get-PSProvider
Get-PSSession
Get-PSSessionConfiguration
Get-PSSnapin
Get-Random
Get-ScheduledJob
Get-ScheduledJobOption
Get-SecureBootPolicy
Get-SecureBootUEFI
Get-Service
Get-Tpm
Get-TraceSource
Get-Transaction
Get-TroubleshootingPack
Get-TypeData
Get-UICulture
Get-Unique
Get-Variable
Get-WheaMemoryPolicy
Get-WinAcceptLanguageFromLanguageList
OptOut
Get-WinCultureFromLanguageListOptOut
Get-WinDefaultInputMethodOverride
Get-WindowsDriver
Get-WindowsEdition
Get-WindowsErrorReporting
Get-WindowsImage
Get-WindowsOptionalFeature
Get-WindowsPackage
Get-WinEvent
Get-WinHomeLocation
Get-WinLanguageBarOption
Get-WinSystemLocale
Get-WinUILanguageOverride
Get-WinUserLanguageList
Get-WmiObject
Get-WSManCredSSP
Get-WSManInstance
Group-Object
Import-Alias
Import-Certificate
Import-Clixml
Import-Counter
Import-Csv
Import-IscsiVirtualDisk
Import-LocalizedData
Import-Module
Import-PfxCertificate
Import-PSSession
Import-TpmOwnerAuth
Initialize-Tpm
Install-NfsMappingStore
Invoke-ASCmd
Invoke-BpaModel
Invoke-CimMethod
Invoke-Command
Invoke-Expression
Invoke-History
Invoke-Item
Invoke-PolicyEvaluation
Invoke-ProcessCube
Invoke-ProcessDimension
Invoke-ProcessPartition
Invoke-RestMethod
Invoke-Sqlcmd
Invoke-TroubleshootingPack
Invoke-WebRequest
Invoke-WmiMethod
Invoke-WSManAction
Join-DtcDiagnosticResourceManager
Join-Path
Join-SqlAvailabilityGroup
Limit-EventLog
Measure-Command
Measure-Object
Merge-Partition
Mount-IscsiVirtualDiskSnapshot
Mount-WindowsImage
Move-Item
Move-ItemProperty
New-Alias
New-AppLockerPolicy
New-CertificateNotificationTask
New-CimInstance
New-CimSession
New-CimSessionOption
New-DtcDiagnosticTransaction
New-Event
New-EventLog
New-IscsiServerTarget
New-IscsiVirtualDisk
New-Item
New-ItemProperty
New-JobTrigger
New-Module
New-ModuleManifest
New-NetIPsecAuthProposal
New-NetIPsecMainModeCryptoProposal
New-NetIPsecQuickModeCryptoProposal
New-NfsMappedIdentity
New-NfsNetgroup
New-Object
New-PSDrive
New-PSSession
New-PSSessionConfigurationFile
New-PSSessionOption
New-PSTransportOption
New-PSWorkflowExecutionOption
New-RestoreFolder
New-RestoreLocation
New-ScheduledJobOption
New-SelfSignedCertificate
New-Service
New-SqlAvailabilityGroup
New-SqlAvailabilityGroupListener
New-SqlAvailabilityReplica
New-SqlHADREndpoint
New-TimeSpan
New-Variable
New-WebServiceProxy
New-WinEvent
New-WinUserLanguageList
New-WSManInstance
New-WSManSessionOption
Out-Default
Out-File
Out-GridView
Out-Host
Out-Null
Out-Printer
Out-String
Pop-Location
Push-Location
Read-Host
Receive-DtcDiagnosticTransaction
Receive-Job
Receive-PSSession
Register-CimIndicationEvent
Register-EngineEvent
Register-ObjectEvent
Register-PSSessionConfiguration
Register-ScheduledJob
Register-WmiEvent
Remove-AppxPackage
Remove-AppxProvisionedPackage
Remove-BitsTransfer
Remove-CertificateEnrollmentPolicyServer
Remove-CertificateNotificationTask
Remove-CimInstance
Remove-CimSession
Remove-Computer
Remove-Event
Remove-EventLog
Remove-IscsiServerTarget
Remove-IscsiVirtualDisk
Remove-IscsiVirtualDiskSnapshot
Remove-IscsiVirtualDiskTargetMapping
Remove-Item

Remove-ItemProperty
Remove-Job
Remove-JobTrigger
Remove-Module
Remove-NfsMappedIdentity
Remove-NfsNetgroup
Remove-PSBreakpoint
Remove-PSDrive
Remove-PSSession
Remove-PSSnapin
Remove-RoleMember
Remove-SqlAvailabilityDatabase
Remove-SqlAvailabilityGroup
Remove-SqlAvailabilityReplica
Remove-TypeData
Remove-Variable
Remove-WindowsDriver
Remove-WindowsPackage
Remove-WmiObject
Remove-WSManInstance
Rename-Computer
Rename-Item
Rename-ItemProperty
Repair-WindowsImage
Reset-ComputerMachinePassword
Resolve-DnsName
Resolve-Path
Restart-Computer
Restart-Service
Restore-ASDatabase
Restore-Computer
Restore-IscsiVirtualDisk
Restore-SqlDatabase
Resume-BitsTransfer
Resume-Job
Resume-Service
Resume-SqlAvailabilityDatabase
Save-Help
Save-WindowsImage
Select-Object
Select-String
Select-Xml
Send-DtcDiagnosticTransaction
Send-MailMessage
Set-Acl
Set-Alias
Set-AppLockerPolicy
Set-AuthenticodeSignature
Set-BitsTransfer
Set-BpaResult
Set-CertificateAutoEnrollmentPolicy
Set-CimInstance
Set-Content
Set-Culture
Set-Date
Set-ExecutionPolicy
Set-IscsiServerTarget
Set-IscsiTargetServerSetting
Set-IscsiVirtualDisk
Set-IscsiVirtualDiskSnapshot
Set-Item
Set-ItemProperty
Set-JobTrigger
Set-KdsConfiguration
Set-Location
Set-NfsMappedIdentity
Set-NfsNetgroup
Set-PSBreakpoint
Set-PSDebug
Set-PSSessionConfiguration
Set-ScheduledJob
Set-ScheduledJobOption
Set-SecureBootUEFI
Set-Service
Set-SqlAvailabilityGroup
Set-SqlAvailabilityGroupListener
Set-SqlAvailabilityReplica
Set-SqlHADREndpoint
Set-StrictMode
Set-TpmOwnerAuth
Set-TraceSource
Set-Variable
Set-WheaMemoryPolicy
Set-WinAcceptLanguageFromLanguageListOptOut
Set-WinCultureFromLanguageListOptOut
Set-WinDefaultInputMethodOverride
Set-WindowsEdition
Set-WindowsProductKey
Set-WinHomeLocation
Set-WinLanguageBarOption
Set-WinSystemLocale
Set-WinUILanguageOverride
Set-WinUserLanguageList
Set-WmiInstance
Set-WSManInstance
Set-WSManQuickConfig
Show-Command
Show-ControlPanelItem
Show-EventLog
Sort-Object
Split-Path
Start-BitsTransfer
Start-DtcDiagnosticResourceManager
Start-Job
Start-Process
Start-Service
Start-Sleep
Start-Transaction
Start-Transcript
Stop-Computer
Stop-DtcDiagnosticResourceManager
Stop-Job
Stop-Process
Stop-Service
Stop-Transcript
Suspend-BitsTransfer
Suspend-Job
Suspend-Service
Suspend-SqlAvailabilityDatabase
Switch-Certificate
Switch-SqlAvailabilityGroup
Tee-Object
Test-AppLockerPolicy
Test-Certificate
Test-ComputerSecureChannel
Test-Connection
Test-KdsRootKey
Test-ModuleManifest
Test-NfsMappedIdentity
Test-Path
Test-PSSessionConfigurationFile
Test-SqlAvailabilityGroup
Test-SqlAvailabilityReplica
Test-SqlDatabaseReplicaState
Test-WSMan
Trace-Command
Unblock-File
Unblock-Tpm
Undo-DtcDiagnosticTransaction
Undo-Transaction
Unregister-Event
Unregister-PSSessionConfiguration
Unregister-ScheduledJob
Update-FormatData
Update-Help
Update-List
Update-TypeData
Use-Transaction
Use-WindowsUnattend
Wait-Event
Wait-Job
Wait-Process
Where-Object
Write-Debug
Write-Error
Write-EventLog
Write-Host
Write-Output
Write-Progress
Write-Verbose
Write-Warning

B. Les fonctions

A:
Add-BCDataCacheExtension
Add-BitLockerKeyProtector

Add-DnsClientNrptRule
Add-DtcClusterTMMapping
Add-InitiatorIdToMaskingSet
Add-NetIPHttpsCertBinding
Add-NetLbfoTeamMember
Add-NetLbfoTeamNic
Add-NetSwitchTeamMember
Add-OdbcDsn
Add-PartitionAccessPath
Add-PhysicalDisk
Add-Printer
Add-PrinterDriver
Add-PrinterPort
Add-RDServer
Add-RDSessionHost
Add-RDVirtualDesktopToCollection
Add-TargetPortToMaskingSet
Add-VirtualDiskToMaskingSet
Add-VpnConnection
B:
Backup-BitLockerKeyProtector
Block-SmbShareAccess
C:
cd..
cd\
Clear-BCCache
Clear-BitLockerAutoUnlock
Clear-Disk
Clear-DnsClientCache
Clear-Host
Close-SmbOpenFile
Close-SmbSession
Connect-IscsiTarget
Connect-VirtualDisk
Copy-NetFirewallRule
Copy-NetIPsecMainModeCryptoSet
Copy-NetIPsecMainModeRule
Copy-NetIPsecPhase1AuthSet
Copy-NetIPsecPhase2AuthSet
Copy-NetIPsecQuickModeCryptoSet
Copy-NetIPsecRule
D:
Disable-BC
Disable-BCDowngrading
Disable-BCServeOnBattery
Disable-BitLocker
Disable-BitLockerAutoUnlock
Disable-DAManualEntryPointSelection
Disable-MMAgent
Disable-NetAdapter
Disable-NetAdapterBinding
Disable-NetAdapterChecksumOffload
Disable-NetAdapterEncapsulatedPacketTaskOffload
Disable-NetAdapterIPsecOffload
Disable-NetAdapterLso
Disable-NetAdapterPowerManagement
Disable-NetAdapterQos
Disable-NetAdapterRdma
Disable-NetAdapterRsc
Disable-NetAdapterRss
Disable-NetAdapterSriov
Disable-NetAdapterVmq
Disable-NetDnsTransitionConfiguration
Disable-NetFirewallRule
Disable-NetIPHttpsProfile
Disable-NetIPsecMainModeRule
Disable-NetIPsecRule
Disable-NetNatTransitionConfiguration
Disable-OdbcPerfCounter
Disable-PhysicalDiskIndication
Disable-PSTrace
Disable-PSWSManCombinedTrace
Disable-RDVirtualDesktopADMachineAccountReuse
Disable-ScheduledTask
Disable-ServerManagerStandardUserRemoting
Disable-Ual
Disable-WdacBidTrace
Disable-WSManTrace
Disconnect-IscsiTarget
Disconnect-NfsSession
Disconnect-RDUser
Disconnect-VirtualDisk
Dismount-DiskImage
E:
Enable-BCDistributed
Enable-BCDowngrading
Enable-BCHostedClient
Enable-BCHostedServer
Enable-BCLocal
Enable-BCServeOnBattery
Enable-BitLocker
Enable-BitLockerAutoUnlock
Enable-DAManualEntryPointSelection
Enable-MMAgent
Enable-NetAdapter
Enable-NetAdapterBinding
Enable-NetAdapterChecksumOffload
Enable-NetAdapterEncapsulatedPacketTaskOffload
Enable-NetAdapterIPsecOffload
Enable-NetAdapterLso
Enable-NetAdapterPowerManagement
Enable-NetAdapterQos
Enable-NetAdapterRdma
Enable-NetAdapterRsc
Enable-NetAdapterRss
Enable-NetAdapterSriov
Enable-NetAdapterVmq
Enable-NetDnsTransitionConfiguration
Enable-NetFirewallRule
Enable-NetIPHttpsProfile
Enable-NetIPsecMainModeRule
Enable-NetIPsecRule
Enable-NetNatTransitionConfiguration
Enable-OdbcPerfCounter
Enable-PhysicalDiskIndication
Enable-PSTrace
Enable-PSWSManCombinedTrace
Enable-RDVirtualDesktopADMachineAccountReuse
Enable-ScheduledTask
Enable-ServerManagerStandardUserRemoting
Enable-Ual
Enable-WdacBidTrace
Enable-WSManTrace
Export-BCCachePackage
Export-BCSecretKey
Export-RDPersonalVirtualDesktopAssignment
Export-ScheduledTask
F:
Format-Volume
G:
Get-AppxLastError
Get-AppxLog
Get-BCClientConfiguration
Get-BCContentServerConfiguration
Get-BCDataCache
Get-BCDataCacheExtension
Get-BCHashCache
Get-BCHostedCacheServerConfiguration
Get-BCNetworkConfiguration
Get-BCStatus
Get-BitLockerVolume
Get-ClusteredScheduledTask
Get-CounterSample
Get-DAClientExperienceConfiguration
Get-DAConnectionStatus
Get-DAEntryPointTableItem
Get-Disk
Get-DiskImage
Get-DisplayResolution
Get-DnsClient
Get-DnsClientCache
Get-DnsClientGlobalSetting
Get-DnsClientNrptGlobal
Get-DnsClientNrptPolicy
Get-DnsClientNrptRule
Get-DnsClientServerAddress
Get-Dtc
Get-DtcAdvancedHostSetting

Get-DtcAdvancedSetting
Get-DtcClusterDefault
Get-DtcClusterTMMapping
Get-DtcDefault
Get-DtcLog
Get-DtcNetworkSetting
Get-DtcTransaction
Get-DtcTransactionsStatistics
Get-DtcTransactionsTraceSession
Get-DtcTransactionsTraceSetting
Get-FileIntegrity
Get-InitiatorId
Get-InitiatorPort
Get-IscsiConnection
Get-IscsiSession
Get-IscsiTarget
Get-IscsiTargetPortal
Get-IseSnippet
Get-LogProperties
Get-MaskingSet
Get-MMAgent
Get-NCSIPolicyConfiguration
Get-Net6to4Configuration
Get-NetAdapter
Get-NetAdapterAdvancedProperty
Get-NetAdapterBinding
Get-NetAdapterChecksumOffload
Get-NetAdapterEncapsulatedPacketTaskOffload
Get-NetAdapterHardwareInfo
Get-NetAdapterIPsecOffload
Get-NetAdapterLso
Get-NetAdapterPowerManagement
Get-NetAdapterQos
Get-NetAdapterRdma
Get-NetAdapterRsc
Get-NetAdapterRss
Get-NetAdapterSriov
Get-NetAdapterSriovVf
Get-NetAdapterStatistics
Get-NetAdapterVmq
Get-NetAdapterVmqQueue
Get-NetAdapterVPort
Get-NetConnectionProfile
Get-NetDnsTransitionConfiguration
Get-NetDnsTransitionMonitoring
Get-NetFirewallAddressFilter
Get-NetFirewallApplicationFilter
Get-NetFirewallInterfaceFilter
Get-NetFirewallInterfaceTypeFilter
Get-NetFirewallPortFilter
Get-NetFirewallProfile
Get-NetFirewallRule
Get-NetFirewallSecurityFilter
Get-NetFirewallServiceFilter
Get-NetFirewallSetting
Get-NetIPAddress
Get-NetIPConfiguration
Get-NetIPHttpsConfiguration
Get-NetIPHttpsState
Get-NetIPInterface
Get-NetIPsecDospSetting
Get-NetIPsecMainModeCryptoSet
Get-NetIPsecMainModeRule
Get-NetIPsecMainModeSA
Get-NetIPsecPhase1AuthSet
Get-NetIPsecPhase2AuthSet
Get-NetIPsecQuickModeCryptoSet
Get-NetIPsecQuickModeSA
Get-NetIPsecRule
Get-NetIPv4Protocol
Get-NetIPv6Protocol
Get-NetIsatapConfiguration
Get-NetLbfoTeam
Get-NetLbfoTeamMember
Get-NetLbfoTeamNic
Get-NetNatTransitionConfiguration
Get-NetNatTransitionMonitoring
Get-NetNeighbor
Get-NetOffloadGlobalSetting
Get-NetPrefixPolicy
Get-NetQosPolicy
Get-NetRoute
Get-NetSwitchTeam
Get-NetSwitchTeamMember
Get-NetTCPConnection
Get-NetTCPSetting
Get-NetTeredoConfiguration
Get-NetTeredoState
Get-NetTransportFilter
Get-NetUDPEndpoint
Get-NetUDPSetting
Get-NfsClientConfiguration
Get-NfsClientgroup
Get-NfsClientLock
Get-NfsMappingStore
Get-NfsMountedClient
Get-NfsNetgroupStore
Get-NfsOpenFile
Get-NfsServerConfiguration
Get-NfsSession
Get-NfsShare
Get-NfsSharePermission
Get-NfsStatistics
Get-OdbcDriver
Get-OdbcDsn
Get-OdbcPerfCounter
Get-OffloadDataTransferSetting
Get-Partition
Get-PartitionSupportedSize
Get-PerformanceCollector
Get-PhysicalDisk
Get-PrintConfiguration
Get-Printer
Get-PrinterDriver
Get-PrinterPort
Get-PrinterProperty
Get-PrintJob
Get-RDAvailableApp
Get-RDCertificate
Get-RDConnectionBrokerHighAvailability
Get-RDDeploymentGatewayConfiguration
Get-RDFileTypeAssociation
Get-RDLicenseConfiguration
Get-RDPersonalVirtualDesktopAssignment
Get-RDPersonalVirtualDesktopPatchSchedule
Get-RDRemoteApp
Get-RDRemoteDesktop
Get-RDServer
Get-RDSessionCollection
Get-RDSessionCollectionConfiguration
Get-RDSessionHost
Get-RDUserSession
Get-RDVirtualDesktop
Get-RDVirtualDesktopCollection
Get-RDVirtualDesktopCollectionConfiguration
Get-RDVirtualDesktopCollectionJobStatus
Get-RDVirtualDesktopConcurrency
Get-RDVirtualDesktopIdleCount
Get-RDVirtualDesktopTemplateExportPath
Get-RDWorkspace
Get-ResiliencySetting
Get-ScheduledTask
Get-ScheduledTaskInfo
Get-ServerBpaResult
Get-ServerClusterName
Get-ServerEvent
Get-ServerFeature
Get-ServerInventory
Get-ServerService
Get-SmbClientConfiguration
Get-SmbClientNetworkInterface
Get-SmbConnection
Get-SmbMapping
Get-SmbMultichannelConnection
Get-SmbMultichannelConstraint
Get-SmbOpenFile
Get-SmbServerConfiguration
Get-SmbServerNetworkInterface
Get-SmbSession
Get-SmbShare
Get-SmbShareAccess
Get-SmbWitnessClient
Get-StorageJob

Get-StoragePool
Get-StorageProvider
Get-StorageReliabilityCounter
Get-StorageSetting
Get-StorageSubSystem
Get-SupportedClusterSizes
Get-SupportedFileSystems
Get-TargetPort
Get-TargetPortal
Get-Ual
Get-UalDailyAccess
Get-UalDailyDeviceAccess
Get-UalDailyUserAccess
Get-UalDeviceAccess
Get-UalDns
Get-UalHyperV
Get-UalOverview
Get-UalServerDevice
Get-UalServerUser
Get-UalSystemId
Get-UalUserAccess
Get-Verb
Get-VirtualDisk
Get-VirtualDiskSupportedSize
Get-Volume
Get-VolumeCorruptionCount
Get-VolumeScrubPolicy
Get-VpnConnection
Get-WdacBidTrace
Get-WindowsDeveloperLicense
Get-WindowsFeature
Grant-NfsSharePermission
Grant-RDOUAccess
Grant-SmbShareAccess
H:
help
Hide-VirtualDisk
I:
Import-BCCachePackage
Import-BCSecretKey
Import-IseSnippet
Import-RDPersonalVirtualDesktopAssignment
ImportSystemModules
Initialize-Disk
Install-Dtc
Install-WindowsFeature
Invoke-AsWorkflow
Invoke-RDUserLogoff
J:
K:
L:
Lock-BitLocker
M:
mkdir
more
Mount-DiskImage
Move-RDVirtualDesktop
Move-SmbWitnessClient
N:
New-DAEntryPointTableItem
New-EapConfiguration
New-IscsiTargetPortal
New-IseSnippet
New-MaskingSet
New-NetAdapterAdvancedProperty
New-NetFirewallRule
New-NetIPAddress
New-NetIPHttpsConfiguration
New-NetIPsecDospSetting
New-NetIPsecMainModeCryptoSet
New-NetIPsecMainModeRule
New-NetIPsecPhase1AuthSet
New-NetIPsecPhase2AuthSet
New-NetIPsecQuickModeCryptoSet
New-NetIPsecRule
New-NetLbfoTeam
New-NetNatTransitionConfiguration
New-NetNeighbor
New-NetQosPolicy
New-NetRoute
New-NetSwitchTeam
New-NetTransportFilter
New-NfsClientgroup
New-NfsShare
New-Partition
New-PSWorkflowSession
New-RDCertificate
New-RDPersonalVirtualDesktopPatchSchedule
New-RDRemoteApp
New-RDSessionCollection
New-RDSessionDeployment
New-RDVirtualDesktopCollection
New-RDVirtualDesktopDeployment
New-ScheduledTask
New-ScheduledTaskAction
New-ScheduledTaskPrincipal
New-ScheduledTaskSettingsSet
New-ScheduledTaskTrigger
New-SmbMapping
New-SmbMultichannelConstraint
New-SmbShare
New-StoragePool
New-StorageSubsystemVirtualDisk
New-VirtualDisk
New-VirtualDiskClone
New-VirtualDiskSnapshot
O:
Open-NetGPO
Optimize-Volume
oss
P:
Pause
prompt
Publish-BCFileContent
Publish-BCWebContent
Q:
R:
Register-ClusteredScheduledTask
Register-DnsClient
Register-IscsiSession
Register-ScheduledTask
Remove-BCDataCacheExtension
Remove-BitLockerKeyProtector
Remove-DAEntryPointTableItem
Remove-DnsClientNrptRule
Remove-DtcClusterTMMapping
Remove-InitiatorId
Remove-InitiatorIdFromMaskingSet
Remove-IscsiTargetPortal
Remove-MaskingSet
Remove-NetAdapterAdvancedProperty
Remove-NetFirewallRule
Remove-NetIPAddress
Remove-NetIPHttpsCertBinding
Remove-NetIPHttpsConfiguration
Remove-NetIPsecDospSetting
Remove-NetIPsecMainModeCryptoSet
Remove-NetIPsecMainModeRule
Remove-NetIPsecMainModeSA
Remove-NetIPsecPhase1AuthSet
Remove-NetIPsecPhase2AuthSet
Remove-NetIPsecQuickModeCryptoSet
Remove-NetIPsecQuickModeSA
Remove-NetIPsecRule
Remove-NetLbfoTeam
Remove-NetLbfoTeamMember
Remove-NetLbfoTeamNic
Remove-NetNatTransitionConfiguration
Remove-NetNeighbor
Remove-NetQosPolicy
Remove-NetRoute
Remove-NetSwitchTeam
Remove-NetSwitchTeamMember
Remove-NetTransportFilter
Remove-NfsClientgroup
Remove-NfsShare
Remove-OdbcDsn
Remove-Partition
Remove-PartitionAccessPath
Remove-PhysicalDisk
Remove-Printer
Remove-PrinterDriver
Remove-PrinterPort
Remove-PrintJob
Remove-RDPersonalVirtualDesktopAssignment
Remove-RDPersonalVirtualDesktopPatchSchedule
Remove-RDRemoteApp

Remove-RDServer
Remove-RDSessionCollection
Remove-RDSessionHost
Remove-RDVirtualDesktopCollection
Remove-RDVirtualDesktopFromCollection
Remove-ServerPerformanceLog
Remove-SmbMapping
Remove-SmbMultichannelConstraint
Remove-SmbShare
Remove-StoragePool
Remove-TargetPortFromMaskingSet
Remove-VirtualDisk
Remove-VirtualDiskFromMaskingSet
Remove-VpnConnection
Rename-DAEntryPointTableItem
Rename-MaskingSet
Rename-NetAdapter
Rename-NetFirewallRule
Rename-NetIPHttpsConfiguration
Rename-NetIPsecMainModeCryptoSet
Rename-NetIPsecMainModeRule
Rename-NetIPsecPhase1AuthSet
Rename-NetIPsecPhase2AuthSet
Rename-NetIPsecQuickModeCryptoSet
Rename-NetIPsecRule
Rename-NetLbfoTeam
Rename-NetSwitchTeam
Rename-NfsClientgroup
Rename-Printer
Repair-FileIntegrity
Repair-VirtualDisk
Repair-Volume
Reset-BC
Reset-DAClientExperienceConfiguration
Reset-DAEntryPointTableItem
Reset-DtcLog
Reset-NCSIPolicyConfiguration
Reset-Net6to4Configuration
Reset-NetAdapterAdvancedProperty
Reset-NetDnsTransitionConfiguration
Reset-NetIPHttpsConfiguration
Reset-NetIsatapConfiguration
Reset-NetTeredoConfiguration
Reset-NfsStatistics
Reset-PhysicalDisk
Reset-StorageReliabilityCounter
Resize-Partition
Resize-VirtualDisk
Resolve-NfsMappedIdentity
Restart-NetAdapter
Restart-PrintJob
Resume-BitLocker
Resume-PrintJob
Revoke-NfsClientLock
Revoke-NfsMountedClient
Revoke-NfsOpenFile
Revoke-NfsSharePermission
Revoke-SmbShareAccess
S:
Save-NetGPO
Send-RDUserMessage
Set-BCAuthentication
Set-BCCache
Set-BCDataCacheEntryMaxAge
Set-BCMinSMBLatency
Set-BCSecretKey
Set-ClusteredScheduledTask
Set-DAClientExperienceConfiguration
Set-DAEntryPointTableItem
Set-Disk
Set-DisplayResolution
Set-DnsClient
Set-DnsClientGlobalSetting
Set-DnsClientNrptGlobal
Set-DnsClientNrptRule
Set-DnsClientServerAddress
Set-DtcAdvancedHostSetting
Set-DtcAdvancedSetting
Set-DtcClusterDefault
Set-DtcClusterTMMapping
Set-DtcDefault
Set-DtcLog
Set-DtcNetworkSetting
Set-DtcTransaction
Set-DtcTransactionsTraceSession
Set-DtcTransactionsTraceSetting
Set-FileIntegrity
Set-InitiatorPort
Set-IscsiChapSecret
Set-LogProperties
Set-MMAgent
Set-NCSIPolicyConfiguration
Set-Net6to4Configuration
Set-NetAdapter
Set-NetAdapterAdvancedProperty
Set-NetAdapterBinding
Set-NetAdapterChecksumOffload
Set-NetAdapterEncapsulatedPacketTaskOffload
Set-NetAdapterIPsecOffload
Set-NetAdapterLso
Set-NetAdapterPowerManagement
Set-NetAdapterQos
Set-NetAdapterRdma
Set-NetAdapterRsc
Set-NetAdapterRss
Set-NetAdapterSriov
Set-NetAdapterVmq
Set-NetConnectionProfile
Set-NetDnsTransitionConfiguration
Set-NetFirewallAddressFilter
Set-NetFirewallApplicationFilter
Set-NetFirewallInterfaceFilter
Set-NetFirewallInterfaceTypeFilter
Set-NetFirewallPortFilter
Set-NetFirewallProfile
Set-NetFirewallRule
Set-NetFirewallSecurityFilter
Set-NetFirewallServiceFilter
Set-NetFirewallSetting
Set-NetIPAddress
Set-NetIPHttpsConfiguration
Set-NetIPInterface
Set-NetIPsecDospSetting
Set-NetIPsecMainModeCryptoSet
Set-NetIPsecMainModeRule
Set-NetIPsecPhase1AuthSet
Set-NetIPsecPhase2AuthSet
Set-NetIPsecQuickModeCryptoSet
Set-NetIPsecRule
Set-NetIPv4Protocol
Set-NetIPv6Protocol
Set-NetIsatapConfiguration
Set-NetLbfoTeam
Set-NetLbfoTeamMember
Set-NetLbfoTeamNic
Set-NetNatTransitionConfiguration
Set-NetNeighbor
Set-NetOffloadGlobalSetting
Set-NetQosPolicy
Set-NetRoute
Set-NetTCPSetting
Set-NetTeredoConfiguration
Set-NetUDPSetting
Set-NfsClientConfiguration
Set-NfsClientgroup
Set-NfsMappingStore
Set-NfsNetgroupStore
Set-NfsServerConfiguration
Set-NfsShare
Set-OdbcDriver
Set-OdbcDsn
Set-Partition
Set-PhysicalDisk
Set-PrintConfiguration
Set-Printer
Set-PrinterProperty
Set-RDActiveManagementServer
Set-RDCertificate
Set-RDClientAccessName
Set-RDConnectionBrokerHighAvailability
Set-RDDatabaseConnectionString
Set-RDDeploymentGatewayConfiguration
Set-RDFileTypeAssociation
Set-RDLicenseConfiguration
Set-RDPersonalVirtualDesktopAssignment

Set-RDPersonalVirtualDesktopPatchSchedule
Set-RDRemoteApp
Set-RDRemoteDesktop
Set-RDSessionCollectionConfiguration
Set-RDSessionHost
Set-RDVirtualDesktopCollectionConfiguration
Set-RDVirtualDesktopConcurrency
Set-RDVirtualDesktopIdleCount
Set-RDVirtualDesktopTemplateExportPath
Set-RDWorkspace
Set-ResiliencySetting
Set-ScheduledTask
Set-SmbClientConfiguration
Set-SmbServerConfiguration
Set-SmbShare
Set-StoragePool
Set-StorageSetting
Set-StorageSubSystem
Set-VirtualDisk
Set-Volume
Set-VolumeScrubPolicy
Set-VpnConnection
Set-VpnConnectionProxy
Show-NetFirewallRule
Show-NetIPsecRule
Show-VirtualDisk
Show-WindowsDeveloperLicenseRegistration
Start-Dtc
Start-DtcTransactionsTraceSession
Start-PerformanceCollector
Start-ScheduledTask
Start-Trace
Stop-Dtc
Stop-DtcTransactionsTraceSession
Stop-PerformanceCollector
Stop-RDVirtualDesktopCollectionJob
Stop-ScheduledTask
Stop-Trace
Suspend-BitLocker
Suspend-PrintJob
Sync-NetIPsecRule
T:
TabExpansion2
Test-Dtc
Test-NfsMappingStore
Test-RDOUAccess
Test-RDVirtualDesktopADMachineAccountReuse
U:
Unblock-SmbShareAccess
Uninstall-Dtc
Uninstall-WindowsFeature
Unlock-BitLocker
Unregister-ClusteredScheduledTask
Unregister-IscsiSession
Unregister-ScheduledTask
Unregister-WindowsDeveloperLicense
Update-Disk
Update-HostStorageCache
Update-IscsiTarget
Update-IscsiTargetPortal
Update-NetIPsecRule
Update-RDVirtualDesktopCollection
Update-SmbMultichannelConnection
Update-StorageProviderCache
V:
W:
Write-DtcTransactionsTraceSession
X:
Y:
Z:

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

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

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

	$b = [string]::join("", $a)
LBound	$a = 1,2,3,4,5,6,7,8,9 $b = $a.getlowerbound(0)
LCase	$a = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" $a = $a.ToLower()
Left	$a="ABCDEFGHIJKLMNOPQRSTUVWXYZ" $a = $a.substring(0,3)
Len	$a = "abcdefghijklmnopqrstuvwxyz" $b = $a.length
Log	$a = [math]::log(100)
LTrim	$a = ".....123456789....." $a = $a.TrimStart()
RTrim	$a = ".....123456789....." $a = $a.TrimEnd()
Trim	$a = ".....123456789....." $a = $a.Trim()
Mid	$a="ABCDEFG" $a = $a.substring(2,3)
Minute	$a =(get-date).minute
Month	$a = get-date -f "MM" $a = [int] (get-date -f "MM")
MonthName	$a = get-date -f "MMMM"
MsgBox	$a = new-object -comobject wscript.shell $b = $a.popup("This is a test",0,"Test Message Box",1)
Now	$a = get-date
Oct	$a = [System.Convert]::ToString(999,8)
Replace	$a = "bxnxnx" $a = $a -replace("x","a")
RGB	$blue = 10 $green= 10 $red = 10 $a = [long] ($blue + ($green * 256) + ($red * 65536))
Right	$a = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" $a = $a.substring($a.length - 9, 9)
Rnd	$a = new-object random $b = $a.next(1,100) $b = $a.next()
Round	$a = [math]::round(45.987654321, 2)
ScriptEngine	$a = (get-host).version
ScriptEngineBuildVersion	$a = (get-host).version.build
ScriptEngineMajorVersion	$a = (get-host).version.major
ScriptEngineMinorVersion	$a = (get-host).version.minor
Second	$a = (get-date).second
Sgn	$a = [math]::sign(-453)
Sin	$a = [math]::sin(45)
Space	$a = " " * 25 $a = $a + "x"
Split	$a = "atl-ws-01,atl-ws-02,atl-ws-03,atl-ws-04" $b = $a.split(",")
Sqr	$a = [math]::sqrt(144)
StrComp	$a = "dog" $b = "DOG" $c = [String]::Compare($a,$b,$True)
String	$a = "=" * 20
StrReverse	$a = "Scripting Guys" for ($i = $a.length - 1; $i -ge 0; $i--) {$b = $b + ($a.substring($i,1))}
Tan	$a = [math]::tan(45)

Time	$a = get-date -displayhint time
TimeSerial	$a = get-date -h 17 -mi 10 -s 45 -displayhint time
TimeValue	$a = [datetime] "1:45 AM"
TypeName	**$a = 55.86768** **$b = $a.GetType().name**
UBound	$a = "a","b","c","d","e" $a.getupperbound(0) $a.length-1
UCase	$a = "abcdefghijklmnopqrstuvwxyz" $a = $a.ToUpper()
WeekdayName	$a = (get-date).dayofweek $a = (get-date "12/25/2007").dayofweek
Year	$a = (get-date).year $a = (get-date "9/15/2005").year
Const Statement	set-variable -name ForReading -value 1 -option constant
Dim Statement	$a = [string]
Execute Statement	$a = "get-date" invoke-expression $a
Function Statement	function multiplynumbers { $args[0] * $args[1] } multiplynumbers 38 99
On Error Statement	$erroractionpreference = "SilentlyContinue" Incidentally, your choices for this variable include: SilentlyContinue Continue (the default value) Inquire Stop
Option Explicit Statement	set-psdebug –strict set-psdebug -off
Private Statement	$Private:a = 5
Public Statement	$Global:a = 199
Randomize Statement	$a = new-object random $b = $a.next()
ReDim Statement	$a = 1,2,3,4,5 $a = $a + 100 $a = $a[0..2]
Set Statement	$a = new-object -comobject Excel.Application $a.visible = $True
Stop Statement	set-psdebug –step set-psdebug -off
Sub Statement	function multiplynumbers { $args[0] * $args[1] } multiplynumbers 38 99
Description Property	$a = $error[0].ToString()
HelpContext Property	$a = $error[0].helplink
HelpFile Property	$a = $error[0].helplink
Number Property	ScriptHalted $error[0].errorrecord
Source Property	$a = $error[0].source
Clear Method	$error[0] = "" $error.clear()
Raise Method	$b = "The file could not be found."; throw $b

XV. 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

XVI. Les modules

A. Le module PackageManagement

https://www.microsoft.com/en-us/download/details.aspx?id=51451

```
Get-Command -Module PackageManagement
Find-PackageProvider
Install-PackageProvider chocolatey
Set-PackageSource -Name chocolatey -Trusted
Get-PackageSource
Find-Package -Name *Adobe* -Source Chocolatey
Install-Package -Name AdobeReader -ProviderName Chocolatey
```

B. Le module BitsTransfer

```
Import-Module BitsTransfer
Start-BitsTransfer -Source "https://owncloud.mon-domaine.fr/index.php/s/xiiI7SmWQzsGV5G/download"
-Destination "C:\temp\mon-fichier.iso.gz"
Start-BitsTransfer -Source "https://owncloud.mon-domaine.fr/index.php/s/xiiI7SmWQzsGV5G/download"
-Destination "C:\temp\mon-fichier.iso.gz" -Asynchronous
Get-BitsTransfer | Complete-BitsTransfer
Get-BitsTransfer -JobId "3a381ff9-2bff-4081-bbff-63eb17abf976" | Complete-BitsTransfer
$MyCred = Get-Credential
Start-BitsTransfer -Source "\\192.168.1.150\Download\BITS\*.*" -Destination "C:\temp\" -
Credential $MyCred
```

https://github.com/florianburnel/PowerShell/blob/master/SYSTEM-Start-BitsDownloadRecursive/Start-BitsDownloadRecursive.ps1

C. Le module PSScriptAnalyzer

https://www.powershellgallery.com/packages/PSScriptAnalyzer/1.12.0

XVII. Téléchargement

```
Invoke-WebRequest -Uri "<lien-du-fichier-a-telecharger" -OutFile "<chemin-vers-fichier-
destination>"
Invoke-WebRequest -Uri "https://github.com/tabad/fusioninventory-agent-windows-
installer/releases/download/2.3.18/fusioninventory-agent_windows-x64_2.3.18.exe" -OutFile
"C:\temp\fusioninventory-agent_windows-x64_2.3.18.exe"
$WebRequest = New-Object System.Net.WebClient
$WebRequest.DownloadFile("<lien-vers-le-fichier>","<chemin-vers-la-destination>")
$WebRequest = New-Object System.Net.WebClient
$WebRequest.DownloadFile("https://github.com/tabad/fusioninventory-agent-windows-
installer/releases/download/2.3.18/fusioninventory-agent_windows-
x64_2.3.18.exe","C:\temp\fusioninventory-agent_windows-x64_2.3.18.exe")
```


XVIII. Exemple de fichier d'aide

```
<#
.SYNOPSIS
    This script geolocation one or several IP address by a web request on the website
geoipview.com

.DESCRIPTION
    Specify a list of IP address that you want to geolocation, and the function return the city
and the country (origin) of the IP.

.PARAMETER IPToCheck
    The list of IP address, it's a required parameter.

.EXAMPLE
    .\Get-IPLocation -IPToCheck "4.4.4.4","8.8.8.8"
    Get the location of two IP address : 4.4.4.4 and 8.8.8.8

.INPUTS

.OUTPUTS

.NOTES
    NAME:    Get-IPLocation.ps1
    AUTHOR:    Florian Burnel
    EMAIL:    florian.burnel@it-connect.fr
    WWW:    www.it-connect.fr
    Twitter:@FlorianBurnel

    VERSION HISTORY:

    1.0     2017.01.17
            Initial Version

#>
```