Administration de Windows

Par défaut, Powershell vient avec la majorité des commandes nécessaires à l'administration de Windows. À quelques exceptions près (entre autres la gestion de Windows Updates)

Utilisateurs et groupes

Récupération d'information sur les utilisateurs et groupes locaux

Get-LocalUser

Une commande simple comme la suivante nous permettrait de vérifier quel utilisateur a un profil local sur une machine ainsi que sa dernière date de connexion. Ce genre d'information est souvent utilisé pour nettoyer les profils locaux d'utilisateurs Active Directory inactifs.

```
PS C:\WINDOWS\system32> Get-LocalUser | Select-Object -Property Name,Enabled,LastLogon

Name               Enabled LastLogon
----               ------- ---------
Administrateur       False
DefaultAccount       False
Info                  True 2025-01-19 19:36:18
Invité               False
WDAGUtilityAccount   False
```

Les informations récupérables par cette commande sont toutefois limitées comme le démontre la commande suivante.

```
PS C:\WINDOWS\system32> Get-LocalUser | Where-Object Name -like Info | Select-Object -Property
*

AccountExpires         :
Description            :
Enabled                : True
FullName               :
PasswordChangeableDate :
PasswordExpires        :
UserMayChangePassword  : True
PasswordRequired       : False
PasswordLastSet        :
LastLogon              : 2025-01-19 19:36:18
Name                   : Info
```

```
SID                   : S-1-5-21-2716239665-1816042345-2264835360-1001
PrincipalSource       : Local
ObjectClass           : Utilisateur
```

Les dates d'expiration et de changement de mot de passe sont habituellement utilisées pour soit alerter les utilisateurs de l'expiration imminente de leur mot de passe ou encore pour effectuer des recommendations en terme de pratique de cybersécurité.

Get-Process

Quoi que Get-Process soit en lien avec l'administration du système plutôt que de ses utilisateurs, elle peut être utilisée pour déterminer quel utilisateur a une session active sur un poste de travail en regardant quelles instances d'un processus commun à tous tel que explorer.exe sont en exécution!

```
PS C:\WINDOWS\system32> Get-Process -IncludeUserName | Where-Object Name -like explorer* |
Select-Object -Property UserName,StartTime

UserName                       StartTime
--------                       ---------
Vidange-Windose\Administrateur 2025-01-19 19:59:14
Vidange-Windose\Info           2025-01-19 18:08:00
```

Get-LocalGroup

La commande Get-LocalGroup nous permet d'interroger le système au sujet des groupes locaux mais tout comme Get-LocalUsers, la commande n'est pas très bavarde.

```
PS C:\WINDOWS\system32> Get-LocalGroup -Name "Administrateurs" | Select-Object -Property *

Description     : Les membres du groupe Administrateurs disposent d’un accès complet et
illimité à l’ordinateur et au
                  domaine
Name            : Administrateurs
SID             : S-1-5-32-544
PrincipalSource : Local
ObjectClass     : Groupe
```

Get-LocalGroupMember

C'est plutôt la commande Get-LocalGroupMember qui nous permet de voir l'association entre les utilisateurs et les groupes.

```
PS C:\WINDOWS\system32> Get-LocalGroupMember -Name "Administrateurs" | Select-Object -Property
*

Name                           SID                                               PrincipalSource
ObjectClass
----                           ---                                               ---------------
-----------
Vidange-Windose\Administrateur S-1-5-21-2716239665-1816042345-2264835360-500           Local
Utilisateur
Vidange-Windose\Info           S-1-5-21-2716239665-1816042345-2264835360-1001          Local
Utilisateur
```

Pour vérifier la liste des groupes auxquels un utilisateur appartient, il faut boucler dans tous les utilisateurs et tous les groupes avec un script tel que celui-ci.

```
PS C:\WINDOWS\system32> Get-LocalUser | ForEach-Object {
>>   $user = $_
>>   return [PSCustomObject]@{
>>     "User" = $user.Name
>>     "Groups" = Get-LocalGroup | Where-Object {
>>       $user.SID -in ($_ | Get-LocalGroupMember | Select-Object -ExpandProperty "SID") |
Select-Object -Property "Name"
>>     }
>>   }
>> }

User               Groups
----               ------
Administrateur     {Administrateurs, Administrateurs Hyper-V, Duplicateurs, IIS_IUSRS...}
DefaultAccount     {Administrateurs, Administrateurs Hyper-V, Duplicateurs, IIS_IUSRS...}
Info               {Administrateurs, Administrateurs Hyper-V, Duplicateurs, IIS_IUSRS...}
Invité             {Administrateurs, Administrateurs Hyper-V, Duplicateurs, IIS_IUSRS...}
WDAGUtilityAccount {Administrateurs, Administrateurs Hyper-V, Duplicateurs, IIS_IUSRS...}
```

Si un utilisateur était mentionné, la première boucle ne serait pas nécessaire.

Gestion des utilisateurs et des groupes locaux

Puisque les objets sont simples, leur création et leur gestion l'est aussi.

New-LocalUser

En se fiant à la commande Get-LocalUser mentionnée plus haut, on peut déterminer les paramètres à saisir pour créer un nouvel utilisateur à l'aide de la commande New-LocalUser.

Si un mot de passe doit être précisé, il doit être mentionné sous forme d'une "SecureString". Pour faciliter la structure de ces commandes, on peut stocker le mot de passe dans une variable plutôt que de faire un "one-liner" mais il est aussi possible de tout effectuer dans la même commande.

```
PS C:\WINDOWS\system32> Get-LocalUser

Name               Enabled Description
----               ------- -----------
Administrateur     True    Compte d’utilisateur d’administration
DefaultAccount     False   Compte utilisateur géré par le système.
Info               True
Invité             False   Compte d’utilisateur invité
WDAGUtilityAccount False   Compte d’utilisateur géré et utilisé par le système pour les
scénarios Windows Defender Applic...

PS C:\WINDOWS\system32> $pwd = ConvertTo-SecureString -String "bobettes" -AsPlainText -Force
PS C:\WINDOWS\system32> New-LocalUser -Name "bob" -FullName "Bobby" -Password $pwd

Name Enabled Description
---- ------- -----------
bob  True

PS C:\WINDOWS\system32> Get-LocalUser

Name               Enabled Description
----               ------- -----------
Administrateur     True    Compte d’utilisateur d’administration
bob                True
DefaultAccount     False   Compte utilisateur géré par le système.
Info               True
Invité             False   Compte d’utilisateur invité
WDAGUtilityAccount False   Compte d’utilisateur géré et utilisé par le système pour les
scénarios Windows Defender Applic...
```

Set-LocalUser

La commande "Set-LocalUser" fonctionne de la même façon que la commande "New-LocalUser" à l'exception que le paramètre "-Name" qui sera mentionné représentera le compte à modifier.

```
PS C:\WINDOWS\system32> Get-LocalUser bob

Name Enabled Description
---- ------- -----------
bob  True


PS C:\WINDOWS\system32> Set-LocalUser -Name bob -Description "Ta mère"
PS C:\WINDOWS\system32> Get-LocalUser bob

Name Enabled Description
---- ------- -----------
bob  True    Ta mère
```

New-LocalGroup

Encore une fois, en se fiant à Get-LocalGroup, on peut déterminer l'ensemble des paramètres qui seront à mentionner à la création d'un groupe.

```
PS C:\WINDOWS\system32> Get-LocalGroup

Name                                Description
----                                -----------
Administrateurs                     Les membres du groupe Administrateurs disposent
d’un accès complet et illimit...
Utilisateurs                        Les utilisateurs ne peuvent pas effectuer de
modifications accidentelles ou i...
Utilisateurs de gestion à distance  Les membres de ce groupe ont accès aux ressources
WMI via des protocoles de g...
Utilisateurs du Bureau à distance   Les membres de ce groupe disposent des droits
nécessaires pour ouvrir une ses...
Utilisateurs OpenSSH                Les membres de ce groupe peuvent se connecter à
cet ordinateur à l’aide de SSH.


PS C:\WINDOWS\system32> New-LocalGroup -Name "Yo Les Jeunes" -Description "Le club secret à
Bob"

Name          Description
----          -----------
Yo Les Jeunes Le club secret à Bob
```

```
PS C:\WINDOWS\system32> Get-LocalGroup

Name                                  Description
----                                  -----------
Yo Les Jeunes                         Le club secret à Bob
Administrateurs                       Les membres du groupe Administrateurs disposent
d’un accès complet et illimit...
Utilisateurs                          Les utilisateurs ne peuvent pas effectuer de
modifications accidentelles ou i...
Utilisateurs de gestion à distance    Les membres de ce groupe ont accès aux ressources
WMI via des protocoles de g...
Utilisateurs du Bureau à distance     Les membres de ce groupe disposent des droits
nécessaires pour ouvrir une ses...
Utilisateurs OpenSSH                  Les membres de ce groupe peuvent se connecter à
cet ordinateur à l’aide de SSH.
```

Set-LocalGroup

Tout comme pour Set-LocalUser, Set-LocalGroup utilise le paramètre "Name" pour déterminer le groupe à modifier.

```
PS C:\WINDOWS\system32> Get-LocalGroup -Name "Yo Les Jeunes"

Name          Description
----          -----------
Yo Les Jeunes Le club secret à Bob



PS C:\WINDOWS\system32> Set-LocalGroup -Name "Yo Les Jeunes" -Description "Le club plus tant
secret à Bob"
PS C:\WINDOWS\system32> Get-LocalGroup -Name "Yo Les Jeunes"

Name          Description
----          -----------
Yo Les Jeunes Le club plus tant secret à Bob
```

Add-LocalGroupMember

Pour attribuer des utilisateurs ou des groupes à un groupe (oui, on peut imbriquer des groupes), la commande Add-LocalGroupMember est aussi simple que de mentionner le nom du membre qui

peut être le nom d'un utilisateur ou d'un groupe et de mentionner le groupe auquel assigner le membre.

On assigne ici l'utilisateur "bob" à son groupe

```
PS C:\WINDOWS\system32> Get-LocalGroupMember "Yo Les Jeunes"
PS C:\WINDOWS\system32> Add-LocalGroupMember -Group "Yo Les Jeunes" -Member "bob"
PS C:\WINDOWS\system32> Get-LocalGroupMember "Yo Les Jeunes"

ObjectClass Name                PrincipalSource
----------- ----                ---------------
Utilisateur Vidange-Windose\bob Local
```

Le principe est le même pour imbriquer un groupe dans un autre

```
PS C:\WINDOWS\system32> Add-LocalGroupMember -Group "Yo Les Jeunes" -Member "Invités"
PS C:\WINDOWS\system32> Get-LocalGroupMember "Yo Les Jeunes"

ObjectClass Name                PrincipalSource
----------- ----                ---------------
Groupe      BUILTIN\Invités     Local
Utilisateur Vidange-Windose\bob Local
```

Remove-LocalGroupMember

Le principe ici est le même que pour l'ajout!

```
PS C:\WINDOWS\system32> Get-LocalGroupMember "Yo Les Jeunes"

ObjectClass Name                PrincipalSource
----------- ----                ---------------
Groupe      BUILTIN\Invités     Local
Utilisateur Vidange-Windose\bob Local



PS C:\WINDOWS\system32> Remove-LocalGroupMember -Group "Yo Les Jeunes" -Member "bob"
PS C:\WINDOWS\system32> Get-LocalGroupMember "Yo Les Jeunes"

ObjectClass Name            PrincipalSource
----------- ----            ---------------
Groupe      BUILTIN\Invités Local
```

Applications et fonctionnalités

Récupération d'information sur les applications et fonctionnalités installées et installables

Get-AppxPackage

La commande Get-AppxPackage vous permet de récupérer les informations sur les applications UWP (Universal Windows Platform) installées sur le système.

```
PS C:\WINDOWS\system32> Get-AppxPackage | Where-Object Name -like *Copilot | Select-Object -
Property *



Name                  : Microsoft.Copilot
Publisher             : CN=Microsoft Corporation, O=Microsoft Corporation, L=Redmond,
S=Washington, C=US
PublisherId           : 8wekyb3d8bbwe
Architecture          : X64
ResourceId            :
Version               : 1.24122.48.0
PackageFamilyName     : Microsoft.Copilot_8wekyb3d8bbwe
PackageFullName       : Microsoft.Copilot_1.24122.48.0_x64__8wekyb3d8bbwe
InstallLocation       : C:\Program
Files\WindowsApps\Microsoft.Copilot_1.24122.48.0_x64__8wekyb3d8bbwe
IsFramework           : False
PackageUserInformation : {}
IsResourcePackage     : False
IsBundle              : False
IsDevelopmentMode     : False
NonRemovable          : False
Dependencies          : {Microsoft.WindowsAppRuntime.1.6_6000.373.1641.0_x64__8wekyb3d8bbwe,
                        Microsoft.VCLibs.140.00.UWPDesktop_14.0.33728.0_x64__8wekyb3d8bbwe,
                        Microsoft.VCLibs.140.00_14.0.33519.0_x64__8wekyb3d8bbwe,
                        Microsoft.Copilot_1.24122.48.0_neutral_split.language-
fr_8wekyb3d8bbwe}
IsPartiallyStaged     : False
```

```
SignatureKind          : Store
Status                 : Ok
```

Get-AppxProvisionedPackage

La commande Get-AppxProvisionedPackage vous permet de récupérer les informations sur les applications UWP (Universal Windows Platform) qui seront déployées pour tous les nouveaux utilisateurs. Cette commande peut aussi être utilisée sur un système hors ligne pour visualiser une image d'installation ou un système qui n'est pas démarré.

```
PS C:\WINDOWS\system32> Get-AppxProvisionedPackage -Online | Where-Object DisplayName -like
MSTeams | Select-Object -Property *



Version          : 24295.605.3225.8804
PackageName      : MSTeams_24295.605.3225.8804_x64__8wekyb3d8bbwe
DisplayName      : MSTeams
PublisherId      : 8wekyb3d8bbwe
MajorVersion     : 24295
MinorVersion     : 605
Build            : 3225
Revision         : 8804
Architecture     : 9
ResourceId       :
InstallLocation  : C:\Program
Files\WindowsApps\MSTeams_24295.605.3225.8804_x64__8wekyb3d8bbwe\AppxManifest.xml
Regions          :
AD;AE;AF;AG;AI;AL;AM;AO;AQ;AR;AS;AT;AU;AW;AX;AZ;BA;BB;BD;BE;BF;BG;BH;BI;BJ;BL;BM;BN;BO;BQ;BR;B
S;BT;B

V;BW;BY;BZ;CA;CC;CD;CF;CG;CH;CI;CK;CL;CM;CO;CR;CU;CV;CW;CX;CY;CZ;DE;DJ;DK;DM;DO;DZ;EC;EE;EG;ER
;ES;ET

;FI;FJ;FK;FM;FO;FR;GA;GB;GD;GE;GF;GG;GH;GI;GL;GM;GN;GP;GQ;GR;GS;GT;GU;GW;GY;HK;HM;HN;HR;HT;HU;
ID;IE;

IL;IM;IN;IO;IQ;IR;IS;IT;JE;JM;JO;JP;KE;KG;KH;KI;KM;KN;KP;KR;KW;KY;KZ;LA;LB;LC;LI;LK;LR;LS;LT;L
U;LV;L

Y;MA;MC;MD;ME;MF;MG;MH;MK;ML;MM;MN;MO;MP;MQ;MR;MS;MT;MU;MV;MW;MX;MY;MZ;NA;NC;NE;NF;NG;NI;NL;NO
;NP;NR
```

```
;NU;NZ;OM;PA;PE;PF;PG;PH;PK;PL;PM;PN;PR;PS;PT;PW;PY;QA;RE;RO;RS;RU;RW;SA;SB;SC;SD;SE;SG;SH;SI;
SJ;SK;

SL;SM;SN;SO;SR;SS;ST;SV;SX;SY;SZ;TC;TD;TF;TG;TH;TJ;TK;TL;TM;TN;TO;TR;TT;TV;TW;TZ;UA;UG;UM;US;U
Y;UZ;V
                  A;VC;VE;VG;VI;VN;VU;WF;WS;XK;YE;YT;ZA;ZM;ZW
Path             :
Online           : True
WinPath          :
SysDrivePath     :
RestartNeeded    : False
LogPath          : C:\WINDOWS\Logs\DISM\dism.log
ScratchDirectory :
LogLevel         : WarningsInfo
```

```
PS F:\> Get-AppxProvisionedPackage -Path E:\mnt | Where-Object DisplayName -like *Outlook* |
Select-Object -Property *



Version          : 1.0.0.0
PackageName      : Microsoft.OutlookForWindows_1.0.0.0_neutral__8wekyb3d8bbwe
DisplayName      : Microsoft.OutlookForWindows
PublisherId      : 8wekyb3d8bbwe
MajorVersion     : 1
MinorVersion     : 0
Build            : 0
Revision         : 0
Architecture     : 11
ResourceId       :
InstallLocation  : %SYSTEMDRIVE%\Program

Files\WindowsApps\Microsoft.OutlookForWindows_1.0.0.0_neutral__8wekyb3d8bbwe\AppxManifest.xml
Regions          :
AD;AE;AF;AG;AI;AL;AM;AO;AQ;AR;AS;AT;AU;AW;AX;AZ;BA;BB;BD;BE;BF;BG;BH;BI;BJ;BL;BM;BN;BO;BQ;BR;B
S;BT;B

V;BW;BY;BZ;CA;CC;CD;CF;CG;CH;CI;CK;CL;CM;CO;CR;CU;CV;CW;CX;CY;CZ;DE;DJ;DK;DM;DO;DZ;EC;EE;EG;ER
;ES;ET
```

```
;FI;FJ;FK;FM;FO;FR;GA;GB;GD;GE;GF;GG;GH;GI;GL;GM;GN;GP;GQ;GR;GS;GT;GU;GW;GY;HK;HM;HN;HR;HT;HU;
ID;IE;

IL;IM;IN;IO;IQ;IR;IS;IT;JE;JM;JO;JP;KE;KG;KH;KI;KM;KN;KP;KR;KW;KY;KZ;LA;LB;LC;LI;LK;LR;LS;LT;L
U;LV;L

Y;MA;MC;MD;ME;MF;MG;MH;MK;ML;MM;MN;MO;MP;MQ;MR;MS;MT;MU;MV;MW;MX;MY;MZ;NA;NC;NE;NF;NG;NI;NL;NO
;NP;NR

;NU;NZ;OM;PA;PE;PF;PG;PH;PK;PL;PM;PN;PR;PS;PT;PW;PY;QA;RE;RO;RS;RU;RW;SA;SB;SC;SD;SE;SG;SH;SI;
SJ;SK;

SL;SM;SN;SO;SR;SS;ST;SV;SX;SY;SZ;TC;TD;TF;TG;TH;TJ;TK;TL;TM;TN;TO;TR;TT;TV;TW;TZ;UA;UG;UM;US;U
Y;UZ;V
                  A;VC;VE;VG;VI;VN;VU;WF;WS;XK;YE;YT;ZA;ZM;ZW
Path             : E:\mnt
Online           : False
WinPath          :
SysDrivePath     :
RestartNeeded    : False
LogPath          : C:\WINDOWS\Logs\DISM\dism.log
ScratchDirectory :
LogLevel         : WarningsInfo
```

Get-WindowsCapability

La commande Get-WindowsCapability nous permet de récupérer les informations sur les fonctionnalités avancées/optionnelles de Windows telles que les outils d'administration de serveurs à distance, les services s'il s'agit d'un serveur windows où les packs de languages installés et disponibles. Cette commande peut aussi être utilisée sur un système hors ligne pour visualiser une image d'installation ou un système qui n'est pas démarré.

```
PS C:\WINDOWS\system32> Get-WindowsCapability -Online | Where-Object Name -like OpenSSH* |
Select-Object -Property *



Name             : OpenSSH.Client~~~~0.0.1.0
State            : Installed
Path             :
Online           : True
WinPath          :
```

```
SysDrivePath     :
RestartNeeded    : False
LogPath          : C:\WINDOWS\Logs\DISM\dism.log
ScratchDirectory :
LogLevel         : WarningsInfo

Name             : OpenSSH.Server~~~~0.0.1.0
State            : NotPresent
Path             :
Online           : True
WinPath          :
SysDrivePath     :
RestartNeeded    : False
LogPath          : C:\WINDOWS\Logs\DISM\dism.log
ScratchDirectory :
LogLevel         : WarningsInfo
```

```
PS F:\> Get-WindowsCapability -Path E:\mnt | Where-Object State -like Installed | Select-
Object -Property Name

Name
----
App.StepsRecorder~~~~0.0.1.0
Browser.InternetExplorer~~~~0.0.11.0
DirectX.Configuration.Database~~~~0.0.1.0
Edge.Webview2.Platform~~~~
Hello.Face.20134~~~~0.0.1.0
Language.Basic~~~en-US~0.0.1.0
Language.Basic~~~fr-CA~0.0.1.0
Language.Basic~~~fr-FR~0.0.1.0
Language.Handwriting~~~fr-FR~0.0.1.0
Language.OCR~~~en-US~0.0.1.0
Language.OCR~~~fr-CA~0.0.1.0
Language.Speech~~~fr-CA~0.0.1.0
Language.TextToSpeech~~~fr-CA~0.0.1.0
MathRecognizer~~~~0.0.1.0
Media.WindowsMediaPlayer~~~~0.0.12.0
[...]
OneCoreUAP.OneSync~~~~0.0.1.0
OpenSSH.Client~~~~0.0.1.0
```

```
VBSCRIPT~~~~
Windows.Client.ProjFS~~~~
Windows.DirectoryServices.ADAM.Client.Content~~~~
Windows.HyperV.OptionalFeature.VirtualMachinePlatform.Client.Disabled~~~~
Windows.Kernel.LA57~~~~0.0.1.0
Windows.SimpleTCP.Content~~~~
Windows.Telnet.Client~~~~
Windows.TerminalServices.AppServerClient~~~~
Windows.TFTP.Client~~~~
Windows.WinOcr~~~~
Windows.WorkFolders.Client~~~~
WMIC~~~~
```

Retrait d'applications et fonctionnalités

Remove-AppxPackage

La commande Remove-AppxPackage peut être utilisée par elle même en mentionnant le nom du package ou conjointement à Get-AppxPackage pour désinstaller un package installé.

```
PS C:\WINDOWS\system32> Get-AppxPackage | Where-Object Name -like *Copilot | Remove-
AppxPackage
PS C:\WINDOWS\system32> Get-AppxPackage | Where-Object Name -like *Copilot
PS C:\WINDOWS\system32>
```

Remove-AppxProvisionedPackage

La commande Remove-AppxProvisionedPackage retire le(s) package(s) de la liste de ceux qui seront installé pour les nouveaux utilisateurs.

```
PS F:\> Get-AppxProvisionedPackage -Path E:\mnt | Where-Object DisplayName -like *Outlook* |
Remove-AppxProvisionedPackage



Path          : E:\mnt
Online        : False
RestartNeeded : False




PS F:\> Get-AppxProvisionedPackage -Path E:\mnt | Where-Object DisplayName -like *Outlook*
```

```
PS F:\>
```

Ces deux commandes pourraient être utilisées en conjonction avec un tableau pour désinstaller une liste d'application dans un script de préparation de nouveau système.

```
$list = @()
$list += '*Teams'
$list += '*BingSearch'
$list += '*BingNews'
$list += '*MicrosoftOfficeHub*'
$list += '*Game*'
$list += '*Gaming*'
$list += '*Solitaire*'
$list += '*Help*'
$list += '*Hub'
$list += '*QuickAssist*'

Foreach ($i in $list) {
Get-AppxPackage | Where-Object Name -like $i | Remove-AppxPackage -AllUsers
}
```

Ajout d'applications et fonctionnalités

Quoi qu'il soit parfois difficile de trouver les packages .msix et .msixbundle, le site suivant permet de générer des liens de téléchargement pour les fichiers d'installation des applications du Microsoft Store.

https://store.rg-adguard.net/

Add-AppxProvisionedPackage

Cette commande est utilisée pour déployer une application à tous les utilisateurs d'un système ou pour injecter une application dans une image d'installation de Windows.

Les commandes suivantes ajouteraient Microsoft PC Manager et sa dépendance à une image d'installation de Windows décapsulée avec DISM.

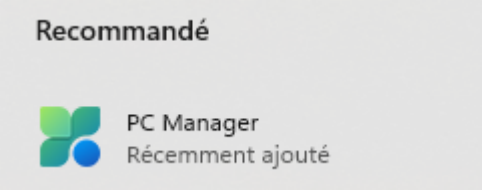
```
PS F:\> Add-AppxProvisionedPackage -PackagePath
F:\Microsoft.WindowsAppRuntime.1.5_5001.373.1736.0_x64__8wekyb3d8bbwe.Msix -Path E:\mnt -
SkipLicense
```

```
Path          : E:\mnt
Online        : False
RestartNeeded : False



PS F:\> Add-AppxProvisionedPackage -PackagePath
F:\Microsoft.MicrosoftPCManager_3.14.18.0_neutral_~_8wekyb3d8bbwe.Msixbundle -Path E:\mnt -
SkipLicense


Path          : E:\mnt
Online        : False
RestartNeeded : False

PS F:\> Get-AppxProvisionedPackage -Path E:\mnt | Where-Object DisplayName -like *Manager*


DisplayName  : Microsoft.MicrosoftPCManager
Version      : 3.14.18.0
Architecture : neutral
ResourceId   : ~
PackageName  : Microsoft.MicrosoftPCManager_3.14.18.0_neutral_~_8wekyb3d8bbwe
Regions      :
```

Les commandes suivantes installeraient Microsoft PC Manager et sa dépendance pour tous les utilisateurs d'un système sur lequel les commandes seraient exécutées.

```
PS C:\Users\Info\Desktop> Add-AppxProvisionedPackage -PackagePath
.\Microsoft.WindowsAppRuntime.1.5_5001.373.1736.0_x64__8wekyb3d8bbwe.Msix -Online -SkipLicense


Path          :
Online        : True
RestartNeeded : False



PS C:\Users\Info\Desktop> Add-AppxProvisionedPackage -PackagePath
.\Microsoft.MicrosoftPCManager_3.14.18.0_neutral_~_8wekyb3d8bbwe.Msixbundle -Online -
SkipLicense
```

```
Path          :
Online        : True
RestartNeeded : False

PS C:\Users\Info\Desktop> Get-AppxProvisionedPackage -Online | Where-Object DisplayName -like
*Manager*



DisplayName  : Microsoft.MicrosoftPCManager
Version      : 3.14.18.0
Architecture : neutral
ResourceId   : ~
PackageName  : Microsoft.MicrosoftPCManager_3.14.18.0_neutral_~_8wekyb3d8bbwe
Regions      :
```

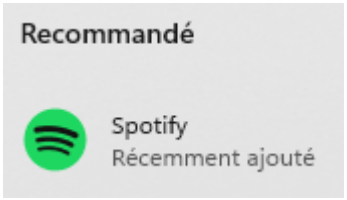


Add-AppxPackage

La commande suivante installerait Spotify et ses dépendances pour l'utilisateur exécutant la commande.

```
PS C:\Users\Info\desktop> Get-Item -Path .\Spotify\* | Add-AppxPackage
PS C:\Users\Info\desktop> Get-AppxPackage | Where-Object Name -like *Spotify* | Select-Object
-Property *



Name                  : SpotifyAB.SpotifyMusic
Publisher             : CN=453637B3-4E12-4CDF-B0D3-2A3C863BF6EF
PublisherId           : zpdnekdrzrea0
Architecture          : X64
ResourceId            :
Version               : 1.255.235.0
PackageFamilyName     : SpotifyAB.SpotifyMusic_zpdnekdrzrea0
PackageFullName       : SpotifyAB.SpotifyMusic_1.255.235.0_x64__zpdnekdrzrea0
InstallLocation       : C:\Program
```

```
Files\WindowsApps\SpotifyAB.SpotifyMusic_1.255.235.0_x64__zpdnekdrzrea0
IsFramework            : False
PackageUserInformation : {}
IsResourcePackage      : False
IsBundle               : False
IsDevelopmentMode      : False
NonRemovable           : False
Dependencies           : {Microsoft.WindowsAppRuntime.1.2_2000.802.31.0_x64__8wekyb3d8bbwe,
                         Microsoft.VCLibs.140.00_14.0.33519.0_x64__8wekyb3d8bbwe,
                         Microsoft.VCLibs.140.00.UWPDesktop_14.0.33728.0_x64__8wekyb3d8bbwe}
IsPartiallyStaged      : False
SignatureKind          : Store
Status                 : Ok
```



Registre Windows

Powershell traite les entrées de registre au même titre que les fichiers et peuvent donc être récupérés, créés et modifiés avec les mêmes commandes que les fichiers. Plutôt que de viser une lettre de lecteur et un répertoire tel que C:\, la "ruche" ou le registre sera visé avec un identifiant comme HKLM:\

Accès aux ruches du registre Windows

Par défaut, seulement HKCU (*HKEY_CURRENT_USER*) et HKLM (*HKEY_LOCAL_MACHINE*) sont accessibles. Puisque le registre est traité au même titre qu'un lecteur, il est possible d'observer ce qui est accessible dans une session ou un script powershell ainsi que de gagner accès à d'autres ruches en manipulant les PSDrives.

Get-PSDrive

Pour observer la liste des ruches de registres accessibles, il est possible d'utiliser la commande "Get-PSDrive" et de filtrer les résultats par "Provider" (type d'objet à manipuler).

```
PS C:\WINDOWS\system32> Get-PSDrive | Where-Object Provider -like *Registry*

Name           Used (GB)     Free (GB) Provider
Root                                            CurrentLocation
----           ---------     --------- --------      ----
                                                ---------------
HKCU                                  Registry      HKEY_CURRENT_USER
HKLM                                  Registry      HKEY_LOCAL_MACHINE
```

New-PSDrive

Il est possible de gagner accès aux autres ruches telles que HKCR (*HKEY_CLASSES_ROOT*) et HKU (*HKEY_USERS*) en les ajoutant comme PSDrive.

```
PS C:\WINDOWS\system32> New-PSDrive -PSProvider Registry -Name HKU -Root HKEY_USERS

Name           Used (GB)     Free (GB) Provider
Root                                            CurrentLocation
----           ---------     --------- --------      ----
                                                ---------------
HKU                                   Registry      HKEY_USERS
```

```
PS C:\WINDOWS\system32> New-PSDrive -PSProvider Registry -Name HKCR -Root HKEY_CLASSES_ROOT

Name           Used (GB)     Free (GB) Provider
Root                                             CurrentLocation
----           ---------     --------- --------      ----
                                                 ---------------
HKCR                                  Registry      HKEY_CLASSES_ROOT

PS C:\WINDOWS\system32> Get-PSDrive | Where-Object Provider -like *Registry*

Name           Used (GB)     Free (GB) Provider
Root                                             CurrentLocation
----           ---------     --------- --------      ----
                                                 ---------------
HKCR                                  Registry      HKEY_CLASSES_ROOT
HKCU                                  Registry      HKEY_CURRENT_USER
HKLM                                  Registry      HKEY_LOCAL_MACHINE
HKU                                   Registry      HKEY_USERS
```

Récupération d'information du registre

Get-ItemProperty

La commande Get-ItemProperty est utilisée pour récupérer l'information sur une valeur unique dans une clé. L'exemple suivant afficherait l'état du paramètre masquant les extensions de fichier connus dans l'explorateur Windows.

```
PS C:\Users\alexa> Get-ItemProperty -Path
"HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced" -Name "HideFileExt"

HideFileExt  : 0
PSPath       :
Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersio
n\Explorer
               \Advanced
PSParentPath :
Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersio
n\Explorer
PSChildName  : Advanced
```

```
PSDrive      : HKCU
PSProvider   : Microsoft.PowerShell.Core\Registry
```

Sa valeur étant 0, les extensions de fichiers connues sont donc affichées dans l'explorateur de fichiers.

Elle peut être utilisée sur une clé sans mentionner de valeur pour afficher toutes les valeurs assignées à une clé. L'exemple suivant afficherait l'ensemble des applications qui démarreront à la connexion de l'utilisateur concerné.

```
PS C:\Users\alexa> Get-ItemProperty -Path
"HKCU:\Software\Microsoft\Windows\CurrentVersion\Run" | Select-Object -Property *


Discord                 : "C:\Users\alexa\AppData\Local\Discord\Update.exe" --processStart
Discord.exe
Steam                   : "C:\Program Files (x86)\Steam\steam.exe" -silent
electron.app.Base Camp™  : C:\Program Files (x86)\Mountain Base Camp\Base Camp.exe --process-
start-args --hidden
com.squirrel.Teams.Teams : C:\Users\alexa\AppData\Local\Microsoft\Teams\Update.exe --
processStart "Teams.exe"
                           --process-start-args "--system-initiated"
EADM                    : "C:\Program Files\Electronic Arts\EA Desktop\EA
Desktop\EALauncher.exe" -silent
EpicGamesLauncher       : "C:\Program Files (x86)\Epic
Games\Launcher\Portal\Binaries\Win64\EpicGamesLauncher.exe"
                           -silent -launchcontext=boot
HASS.Agent              : C:\Users\alexa\AppData\Roaming\LAB02
Research\HASS.Agent\HASS.Agent.exe
PSPath                  :
Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVers
                           ion\Run
PSParentPath            :
Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVers
                           ion
PSChildName             : Run
PSDrive                 : HKCU
PSProvider              : Microsoft.PowerShell.Core\Registry
```

Get-ChildItem

La commande Get-ChildItem permet de récupérer tous les éléments contenus dans une clé de registre et peut être exécutée récursivement pour aussi afficher les sous-clés. L'exemple suivant

afficherait toutes les clés contenant les paramètres associés au logiciel 7-Zip ainsi que les valeurs associées à ces clés.

```
PS C:\Users\alexa> Get-ChildItem -Path "HKCU:\Software\7-Zip" -Recurse


    Hive: HKEY_CURRENT_USER\Software\7-Zip

Name                           Property
----                           --------
FM                             FolderShortcuts      : {}
                               FolderHistory        : {73, 0, 58, 0...}
                               PanelPath0           : I:\Documents\420-153-AT -
Conception\A2024\TP2_A24\
                               FlatViewArc0         : 0
                               PanelPath1           :
                               FlatViewArc1         : 0
                               ListMode             : 771
                               Position             : {162, 1, 0, 0...}
                               Panels               : {1, 0, 0, 0...}
                               ShowDots             : 1
                               ShowRealFileIcons    : 1
                               FullRow              : 1
                               ShowGrid             : 0
                               SingleClick          : 0
                               AlternativeSelection : 0
                               ShowSystemMenu       : 1
                               CopyHistory          : {70, 0, 58, 0...}


    Hive: HKEY_CURRENT_USER\Software\7-Zip\FM

Name                           Property
----                           --------
Columns                        RootFolder     : {1, 0, 0, 0...}
                               7-Zip.zip      : {1, 0, 0, 0...}
                               FSFolder       : {1, 0, 0, 0...}
                               7-Zip.Rar5     : {1, 0, 0, 0...}
                               7-Zip.Rar      : {1, 0, 0, 0...}
                               7-Zip.gzip     : {1, 0, 0, 0...}
                               7-Zip.Compound : {1, 0, 0, 0...}
                               7-Zip.7z       : {1, 0, 0, 0...}
```

```
                                7-Zip.tar      : {1, 0, 0, 0...}
                                7-Zip.Nsis     : {1, 0, 0, 0...}


    Hive: HKEY_CURRENT_USER\Software\7-Zip

Name                           Property
----                           --------
Options                        MenuIcons    : 1
                               CascadedMenu : 1
                               ContextMenu  : 2147487782
```

Création et définition de valeurs dans le registre

New-ItemProperty

La commande Set-ItemProperty permet de définir une valeur associée à une clé. La commande suivante activerait l'affichage avancé à l'écran de connexion d'un utilisateur (chaque étape plutôt que seulement "Veuillez patienter").

```
New-ItemProperty -Path "HKLM:\Software\Microsoft\Windows\CurrentVersion\Policies\System" -Name
"VerboseStatus" -Value 1
```

New-Item

La commande New-Item permet la création de nouvelles clés. Accompagné du paramètre -Force, la commande créera l'ensemble de la hiérarchie jusqu'à l'item créé mais si la clé était déjà existante, elle sera vidée. Il est donc généralement préférable de s'assurer de son existance avant de tenter sa création. Ceci peut facilement être réalisé à l'aide d'un "try catch" comme suit. Dans cet exemple, _Lab n'existe pas.

```
PS C:\Users\alexa> try {
>>     Get-Item -Path "HKCU:\_Lab" -ErrorAction Stop
>> }
>> catch {
>>     New-Item -Path "HKCU:\_Lab" -Force
>> }



    Hive: HKEY_CURRENT_USER



Name                           Property
```

```
----                           --------
_Lab
```

Dans cet exemple, _Lab existe et contient la valeur "Soleil123" nommée "Password". L'item n'est par conséquent pas remplacé.

```
PS C:\Users\alexa> try {
>>     Get-Item -Path "HKCU:\_Lab" -ErrorAction Stop
>> }
>> catch {
>>     New-Item -Path "HKCU:\_Lab" -Force
>> }



    Hive: HKEY_CURRENT_USER



Name                           Property
----                           --------
_Lab                           Password : Soleil123
```

Set-ItemProperty

La commande Set-ItemProperty permet de modifier une valeur assignée à une clé. Il est possible d'exécuter ceci conjointement à une vérification de l'existance de la clé pour s'assurer de la réussite de l'exécution de la commande comme suit. Dans cet exemple, le bouton de Windows Copilot sera retirée de la barre des tâches de l'utilisateur concerné.

```
$registryPath = "HKCU:\Software\Policies\Microsoft\Windows\WindowsCopilot" 
$registryValueName = "ShowCopilotButton"
$registryValueData = 0
try {
    Get-Item -Path $registryPath -ErrorAction Stop
}
catch {
    New-Item -Path $registryPath -Force
}
Set-ItemProperty -Path $registryPath -Name $registryValueName -Value $registryValueData
```