Manipulation d'objets et commandes communes

Get-Help

La commande Get-Help peut vous permettre au même titre que "man" en Linux de voir tous les paramètres qu'une commande Powershell peut accepter. À sa première exécution, l'aide sera téléchargée pour une panoplie de modules et sera ensuite disponible même hors ligne.

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


NOM
    Get-Help

RÉSUMÉ
    Displays information about PowerShell commands and concepts.

SYNTAXE
    Get-Help [[-Name] <System.String>] [-Category {Alias | Cmdlet | Provider | General | FAQ |
Glossary | HelpFile |
    ScriptCommand | Function | Filter | ExternalScript | All | DefaultHelp | Workflow |
DscResource | Class |
    Configuration}] [-Component <System.String[]>] -Detailed [-Functionality
<System.String[]>] [-Path <System.String>]
    [-Role <System.String[]>] [<CommonParameters>]

    Get-Help [[-Name] <System.String>] [-Category {Alias | Cmdlet | Provider | General | FAQ |
Glossary | HelpFile |
    ScriptCommand | Function | Filter | ExternalScript | All | DefaultHelp | Workflow |
DscResource | Class |
    Configuration}] [-Component <System.String[]>] -Examples [-Functionality
<System.String[]>] [-Path <System.String>]
    [-Role <System.String[]>] [<CommonParameters>]

    Get-Help [[-Name] <System.String>] [-Category {Alias | Cmdlet | Provider | General | FAQ |
```

```
Glossary | HelpFile |
    ScriptCommand | Function | Filter | ExternalScript | All | DefaultHelp | Workflow |
DscResource | Class |
    Configuration}] [-Component <System.String[]>] [-Full] [-Functionality <System.String[]>]
[-Path <System.String>]
    [-Role <System.String[]>] [<CommonParameters>]

    Get-Help [[-Name] <System.String>] [-Category {Alias | Cmdlet | Provider | General | FAQ |
Glossary | HelpFile |
    ScriptCommand | Function | Filter | ExternalScript | All | DefaultHelp | Workflow |
DscResource | Class |
    Configuration}] [-Component <System.String[]>] [-Functionality <System.String[]>] -Online
[-Path <System.String>]
    [-Role <System.String[]>] [<CommonParameters>]

    Get-Help [[-Name] <System.String>] [-Category {Alias | Cmdlet | Provider | General | FAQ |
Glossary | HelpFile |
    ScriptCommand | Function | Filter | ExternalScript | All | DefaultHelp | Workflow |
DscResource | Class |
    Configuration}] [-Component <System.String[]>] [-Functionality <System.String[]>] -
Parameter <System.String> [-Path
    <System.String>] [-Role <System.String[]>] [<CommonParameters>]

    Get-Help [[-Name] <System.String>] [-Category {Alias | Cmdlet | Provider | General | FAQ |
Glossary | HelpFile |
    ScriptCommand | Function | Filter | ExternalScript | All | DefaultHelp | Workflow |
DscResource | Class |
    Configuration}] [-Component <System.String[]>] [-Functionality <System.String[]>] [-Path
<System.String>] [-Role
    <System.String[]>] -ShowWindow [<CommonParameters>]
```

La commande peut aussi vous présenter des exemples d'utilisation de la commande recherchée :

```
PS C:\WINDOWS\system32> Get-Help Get-Help -examples

NOM
    Get-Help

RÉSUMÉ
    Displays information about PowerShell commands and concepts.
```

```
    --- Example 1: Display basic help information about a cmdlet ---

    Get-Help Format-Table
    Get-Help -Name Format-Table
    Format-Table -?

    `Get-Help <cmdlet-name>` is the simplest and default syntax of `Get-Help` cmdlet. You can
omit the Name parameter.

    The syntax `<cmdlet-name> -?` works only for cmdlets.
```

New-Object

La commande New-Object est généralement utilisée pour déclarer un objet et le stocker dans une variable. On peut considérer les objets en Powershell comme une instanciation de classe en programmation. Un bon exemple de ceci est présent dans la section ACL des types d'objets.

Un autre cas d'utilisation commun de cette commande est pour créer un objet de type crédentiels pour éviter de redemander à l'utilisateur ses informations à chaque commande qui les nécessite. Quoi qu'il soit possible de les récupérer avec la commande Get-Credential, si on désire rester dans l'invite de commande ou charger ce type d'objet depuis un fichier, il devient plus simple d'utiliser la commande New-Object de la façon suivante :

```
# Lecture des informations d'authentification
$uname = Read-Host -Prompt "Nom d'utilisateur"
$pwd = Read-Host -Prompt "Mot de passe" -AsSecureString

# Création de l'objet
$credential = New-Object System.Management.Automation.PSCredential($uname,$pwd)
```

> Quoi qu'il soit possible d'inscrire directement des informations d'authentification dans le script, ces scripts sont généralement lisible par les postes de travail et voir même les utilisateurs. Ceci est fortement déconseillé!

New

D'autres commandes "New" servent à créer des objets à pour système d'exploitation tel que des utilisateurs locaux, des groupes locaux, des entrées dans le journal d'évènement, etc.

Par exemple, la commande New-User ajouterait un utilisateur local à votre système :

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

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

PS C:\WINDOWS\system32> $pwd = ConvertTo-SecureString "Soleil123" -AsPlainText -Force
PS C:\WINDOWS\system32> New-LocalUser -FullName "Ta mère" -Name "rita" -Password $pwd -
PasswordNeverExpires -AccountNeverExpires

Name Enabled Description
---- ------- -----------
rita True

PS C:\WINDOWS\system32> Get-LocalUser

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

Add

Les commandes débutant par Add servent généralement à ajouter des entrées à des systèmes existants. Il peut s'agir de créer des fichiers, des entrées de registre, des utilisateurs dans un domaine Active Directory, des fonctionnalités Windows, des périphériques réseau, etc.

Une de ces commandes pourrait servir à ajouter un utilisateur à un groupe.

```
PS C:\WINDOWS\system32> Get-LocalGroupMember -Group "Administrateurs"

ObjectClass Name                            PrincipalSource
----------- ----                            ---------------
Utilisateur DESKTOP-D67ERFV\Administrateur Local
Utilisateur DESKTOP-D67ERFV\Bob            Local

PS C:\WINDOWS\system32> Add-LocalGroupMember -Group "Administrateurs" -Member rita
PS C:\WINDOWS\system32> Get-LocalGroupMember -Group "Administrateurs"

ObjectClass Name                            PrincipalSource
----------- ----                            ---------------
Utilisateur DESKTOP-D67ERFV\Administrateur Local
Utilisateur DESKTOP-D67ERFV\Bob            Local
Utilisateur DESKTOP-D67ERFV\rita           Local
```

Set

Les commandes "Set" sont utilisées pour modifier ou définir un ou des paramètres liés à un objet. Ces commandes remplacent la valeur mentionnée.

Une de ces commandes pourrait servir à modifier le mot de passe d'un utilisateur.

```
PS C:\WINDOWS\system32> $pwd = ConvertTo-SecureString "123Soleil" -AsPlainText -Force
PS C:\WINDOWS\system32> Set-LocalUser -Name rita -Password $pwd
```

Get

Les commandes "Get" servent à consulter les objets système ou les objets de scripts comme les variables. Elles sont souvent utilisées conjointement aux commandes "Select-Object", "Where-Object" et à des commandes d'action telles que les commandes "Set" par du "pipelining".

En voici quelques exemples :

```
# Afficher seulement certaines propriétés déterminées par Select-Object de certains objets
correspondant à un critère déterminé par Where-Object
PS C:\Users\alexa> Get-ChildItem -Path D:\Item | Where-Object Name -like "*.txt" | Select-
Object -Property FullName,LastWriteTime

FullName               LastWriteTime
```

```
--------                ------------
D:\Item\ChildItem-1.txt 2025-01-18 4:58:47 PM
D:\Item\ChildItem-2.txt 2025-01-18 4:38:23 PM

# Renommer les fichiers correspondants à ceux correspondant à la requête Where-Object à l'aide de la commande Rename-Item
PS C:\Users\alexa> Get-ChildItem -Path D:\Item | Where-Object Name -like "*.txt" | Rename-Item -NewName {$_.name -replace 'ChildItem','ObjetEnfant'}
PS C:\Users\alexa> Get-ChildItem -Path D:\Item | Where-Object Name -like "*.txt" | Select-Object -Property FullName,LastWriteTime

FullName                  LastWriteTime
--------                  ------------
D:\Item\ObjetEnfant-1.txt 2025-01-18 4:58:47 PM
D:\Item\ObjetEnfant-2.txt 2025-01-18 4:38:23 PM

# Supprimer les fichiers correspondant au filtre Where-Object
PS C:\Users\alexa> Get-ChildItem -Path D:\Item | Where-Object Name -like "*.txt" | Remove-Item
PS C:\Users\alexa> Get-ChildItem -Path D:\Item


    Directory: D:\Item

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        2025-01-18   4:40 PM                Subdirectory
```

Where-Object

La commande Where-Object généralement utilisée avec une commande de type "Get" permet de filtrer les résultats affichés en fonction d'un ou plusieurs paramètres de l'objet ainsi que les objets résultants sur lesquels la commande "pipelinée" suivante sera appliquée. Dans l'exemple précédent, le filtre récupérait uniquement les fichiers texte donc l'action de retrait n'a pas affecté la structure de dossiers.

Dans l'exemple suivant, seulement

Select-Object

La commande Select-Object permet de déterminer quels paramètres récupérer ou afficher d'un objet. Dans l'exemple précédent, Seulement le nom complet du fichier ainsi que la dernière date

de modification étaient sélectionnés et affichés.

Read-Host/Write-Host

Ces commandes sont généralement utilisées pour questionner l'utilisateur et stocker la réponse dans une variable et pour écrire de la journalisation dans la console.

Revision #14
Created 2025-01-18 21:06:29 UTC by Alexandre Arsenault-Jetté
Updated 2025-01-25 00:44:28 UTC by Alexandre Arsenault-Jetté