

Tutoriels

- [Tester une connexion en PowerShell avec Test-NetConnection](#)
- [Créer un fichier texte et écrire dedans](#)
- [Importer et exporter des drivers](#)
- [Afficher le contenu d'un fichier texte](#)
- [PowerShell : gérer les applications sur Windows avec WinGet](#)

Tester une connexion en PowerShell avec Test-NetConnection

Dans ce tutoriel, je vais vous parler de la Cmdlet PowerShell `Test-NetConnection` qui va vous permettre de lancer des diagnostics réseaux.

Cette commande peut remplacer les utilitaires suivants :

- Ping
- Telnet
- Traceroute (tracert)

Avec une seule Cmdlet PowerShell vous allez pouvoir tout faire !

Dans ce tutoriel voici ce que l'on va voir :

- Comment faire un ping en PowerShell
- Comment faire un « telnet » en PowerShell pour tester un port (TCP)
- Comment faire un trace route en PowerShell

Avant de commencer, la Cmdlet `Test-NetConnection` prend un premier paramètre `-ComputerName` qui n'est pas obligé d'être nommé.

Faire un ping en PowerShell avec Test-NetConnection

On va commencer avec une utilisation simple, qui va être de faire un ping.

```
Test-NetConnection google.fr
```

image.png

Comme on peut le voir sur la capture, la commande nous retourne ces informations :

- Adresse ip distante résolu quand on test un nom de domaine
- L'interface et l'adresse IP source
- Le résultat du ping
- Le temps de latence

Maintenant, l'intérêt de PowerShell c'est que l'on va pouvoir faire des scripts, pour illustrer cela via un script simple qui nous permet de tester un ping.

```
<#  
□Script for TEST Connection  
#>  
  
param(  
□$RemoteHost = "google.fr"  
)  
  
$ResultTest = Test-NetConnection -ComputerName $RemoteHost  
  
if( $ResultTest.PingSucceeded ){  
□Write-Output "Test OK for $RemoteHost"  
}else{  
  
□Write-Output "/!\ Test KO for $RemoteHost"  
}
```

Pour récupérer le résultat de la commande, il faut tester PingSucceeded.

Voici le script le résultat du script avec un hôte qui répond et qui ne répond pas.

image.png

Si vous souhaitez faire plusieurs ping ou un ping en continu, il faut utiliser la Cmdlet `Test-Connection`.

Pour effectuer 100 ping

```
Test-Connection google.fr -Count 100
```

Pour un ping en continu :

```
Test-Connection google.fr -Count ([int32]::MaxValue)
```

Tester un port en PowerShell avec Test-NetConnection

Maintenant, on va voir comment tester un port en PowerShell comme on le ferait avec [telnet](#).

L'avantage d'utiliser `Test-NetConnection` par rapport à telnet, c'est qu'il n'y a pas besoin d'installer le client Telnet.

Pour tester un port, il faut ajouter le paramètre `-Port` et le numéro de celui.

Ce qui nous donne :

```
Test-NetConnection rdr-it.com -Port 443
```

image.png

Ici, j'ai testé le port 443 (HTTPS) du serveur du site Internet, pour avoir le résultat du test, il faut regarder la valeur de `TcpTestSucceeded`.

En cas d'échec, la Cmdlet avec tester l'hôte à l'aide d'un ping.

image.png

On peut aussi exploiter ce retour dans un script PowerShell, à la différence du ping, il faut analyser le résultat de `TcpTestSucceeded`.

Tester la route avec Test-NetConnection

Pour finir ce tutoriel, nous allons voir comment tester le routage avec `Test-NetConnection` en PowerShell.

Si vous avez plusieurs routes différentes de configurer, le premier paramètre que l'on peut utiliser est `-DiagnoseRouting`, qui va analysé quel route (passerelle) va être utiliser pour accéder au serveur distant.

image.png

Le second paramètre que l'on va voir est `-tracroute` qui permet d'avoir l'équivalent d'un `tracert` et de voir tous les routeurs qui sont utilisés.

image.png

Créer un fichier texte et écrire dedans

Dans ce « petit » tutoriel, je vais vous expliquer comment créer un fichier texte en Powershell et ensuite comment écrire du contenu dedans.

Dans ce tutoriel, nous allons voir 4 Cmdlet Powershell :

- New-item : qui va être utilisé pour créer le fichier text.
- Set-Content : qui permet d'écrire dans le fichier.
- Add-Content : permet d'ajouter du contenu
- Get-Content : qui affiche le contenu du fichier

Nous allons commencer par créer le fichier.

```
New-Item TextFile.txt
```

image.png

Ici le fichier est créé dans le dossier courant, il est possible de mettre un chemin absolu.

La création d'un fichier est très simple et se fait simplement avec la Cmdlet New-Item en indiquant le chemin / nom du fichier.

Maintenant, on va voir comment ajouter du contenu avec les Cmdlet Set-Content et Add-Content. La principale différence entre les deux commandes, c'est que Set-Content va d'abord effacer le fichier puis ajouter le contenu, alors que la commande Add-Content va ajouter le contenu à la fin du fichier.

Pour ajouter du contenu dans un fichier vide ou remplacer le contenu :

```
Set-Content TextFile.txt "Content to add in the text file"
```

image.png

La commande n'a pas de retour particulier.

Pour ajouter du contenu à la fin du fichier :

```
Add-Content TextFile.txt "Content to add in the text file"
```

[image.png](#)

En utilisant la cmdlet Get-Content, on peut voir le contenu du fichier :

```
Get-Content TextFile.txt
```

[image.png](#)

Importer et exporter des drivers

[powershell.png](#)

Il est possible d'exporter les drivers d'un poste pour les sauvegarder et les exporter vers un autre poste.

Export

Après avoir ouvert un terminal PowerShell en tant qu'administrateur, exécutez la commande suivante :

```
Export-WindowsDriver -Online -Destination D:\DriversBackup
```

Le répertoire de destination doit exister et être accessible en écriture.

Import

Après avoir ouvert un terminal PowerShell en tant qu'administrateur, exécutez la commande suivante :

```
Get-ChildItem "D:\DriversBackup\" -Recurse -Filter "*.inf" | ForEach-Object { PNPUtil.exe  
/add-driver $_.FullName /install }
```

Le répertoire source doit contenir l'ensemble des drivers que l'on souhaite importer.

Afficher le contenu d'un fichier texte

Dans ce tutoriel, je vais vous montrer comment afficher le contenu d'un fichier texte (txt, yaml, php, vbs ...) dans une fenêtre PowerShell.

En faite, c'est très simple, il suffit d'utiliser la Cmdlet `Get-Content` et d'indiquer le nom du fichier.

Ce qui donne :

```
Get-Content MyFile.txt
```

image.png

Si vous êtes habitué à la commande `cat` sous Linux, vous pouvez aussi l'utiliser

```
cat MyFile.txt
```

La commande `cat` est ici un alias de la commande `Get-Content`, on peut le voir avec la Cmdlet `Get-Alias`.

image.png

Il est aussi possible de passer des paramètres supplémentaires à la commande pour afficher seulement les x premières ou dernières lignes.

Pour rentre plus lisible, j'ai ajouté des numéros à mon fichier, voici son contenu :

image.png

Comme on peut le voir, le fichier contient 8 lignes.

Pour afficher les x premières lignes, on va ajouter le parametres `-TotalCount X`.

Pour afficher les 3 premières d'un fichier, on va utiliser la commande :

```
Get-Content MyFile.txt -TotalCount 3
```

image.png

Si vous souhaitez afficher les dernières lignes d'un fichier, il faut utiliser le paramètre `-Tail X`.

Ce qui nous donne pour les 4 dernières ligne du fichier :

```
Get-Content MyFile.txt -Tail 4
```

image.png

PowerShell : gérer les applications sur Windows avec WinGet

WinGet est le gestionnaire de paquets pour Windows. Il permet d'installer, de mettre à jour et de gérer des applications via la ligne de commande. Pour simplifier encore plus ces tâches, Microsoft propose le module PowerShell "**Microsoft.WinGet.Client**". Ce module offre des commandes natives PowerShell, rendant l'utilisation de WinGet plus intuitive et puissante, qu'à partir du jeu de commandes de l'outil en lui-même.

Dans ce **tutoriel**, nous allons voir **comment installer le module Microsoft.WinGet.Client** et **l'utiliser pour gérer vos applications sur Windows**, mais également découvrir **certains de ses avantages**.

“ **Note** : je vous recommande d'utiliser PowerShell 7 ou supérieur afin de vous assurer de la compatibilité des commandes qui composent ce tutoriel.

Installation du module Microsoft.WinGet.Client

Ouvrez une console PowerShell sur votre machine et installez le module via la commande suivante :

```
Install-PSResource Microsoft.WinGet.Client
```

Vous pouvez également utiliser l'ancienne commande : **Install-Module**.

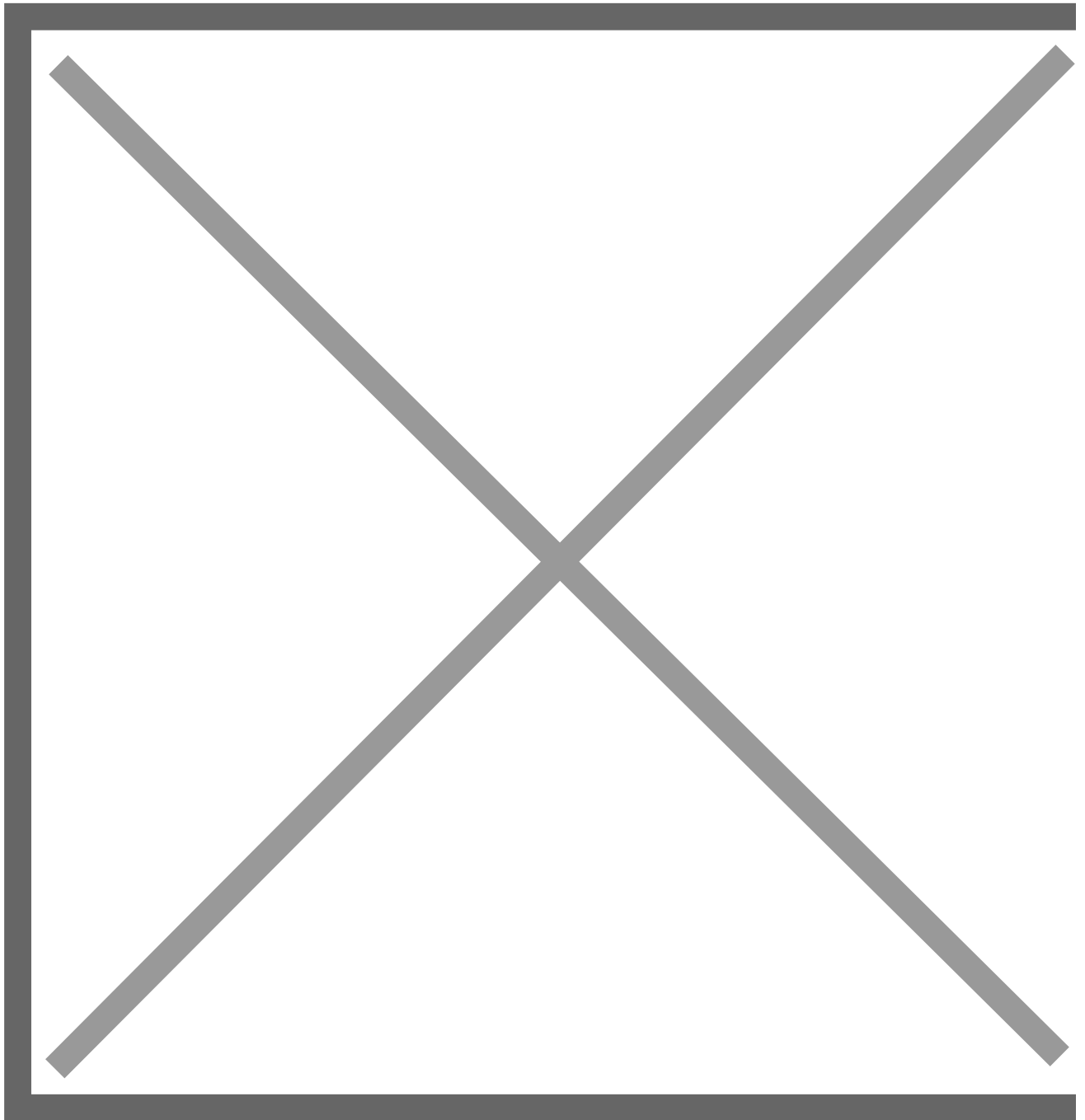
Ensuite, importez le module :

```
Import-Module Microsoft.WinGet.Client
```

Vous pouvez aussi lister les commandes du module :

```
Get-Command -Module Microsoft.WinGet.Client
```

Vous obtiendrez ceci au final :



Quand c'est fait, vous êtes prêt pour la suite !

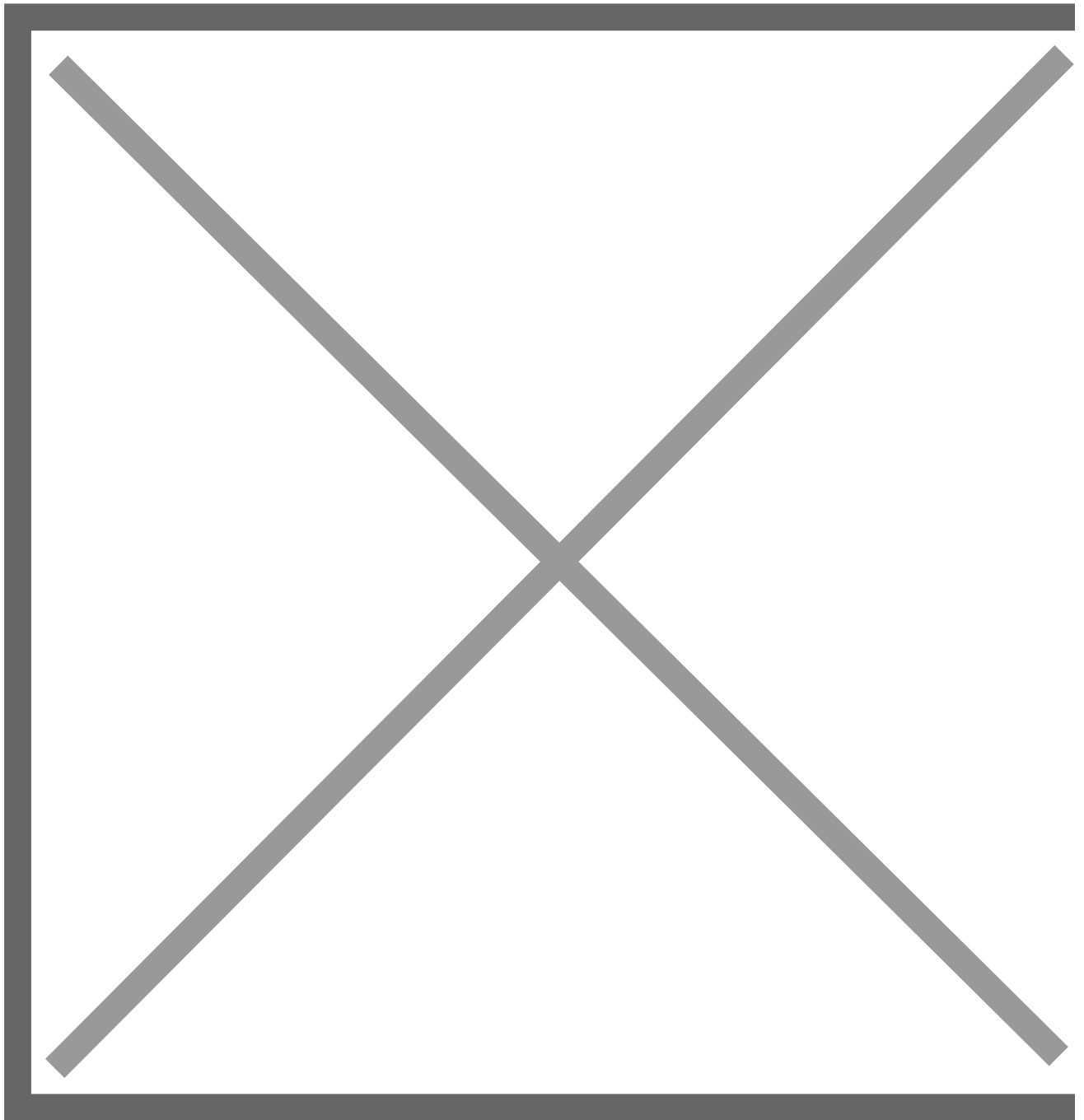
Utilisation du module WinGet

Une fois le module installé et importé, vous pouvez commencer à utiliser les commandes natives PowerShell pour gérer vos applications.

Rechercher un paquet

(exemple : 7-Zip)

```
Find-WinGetPackage "7-Zip"
```

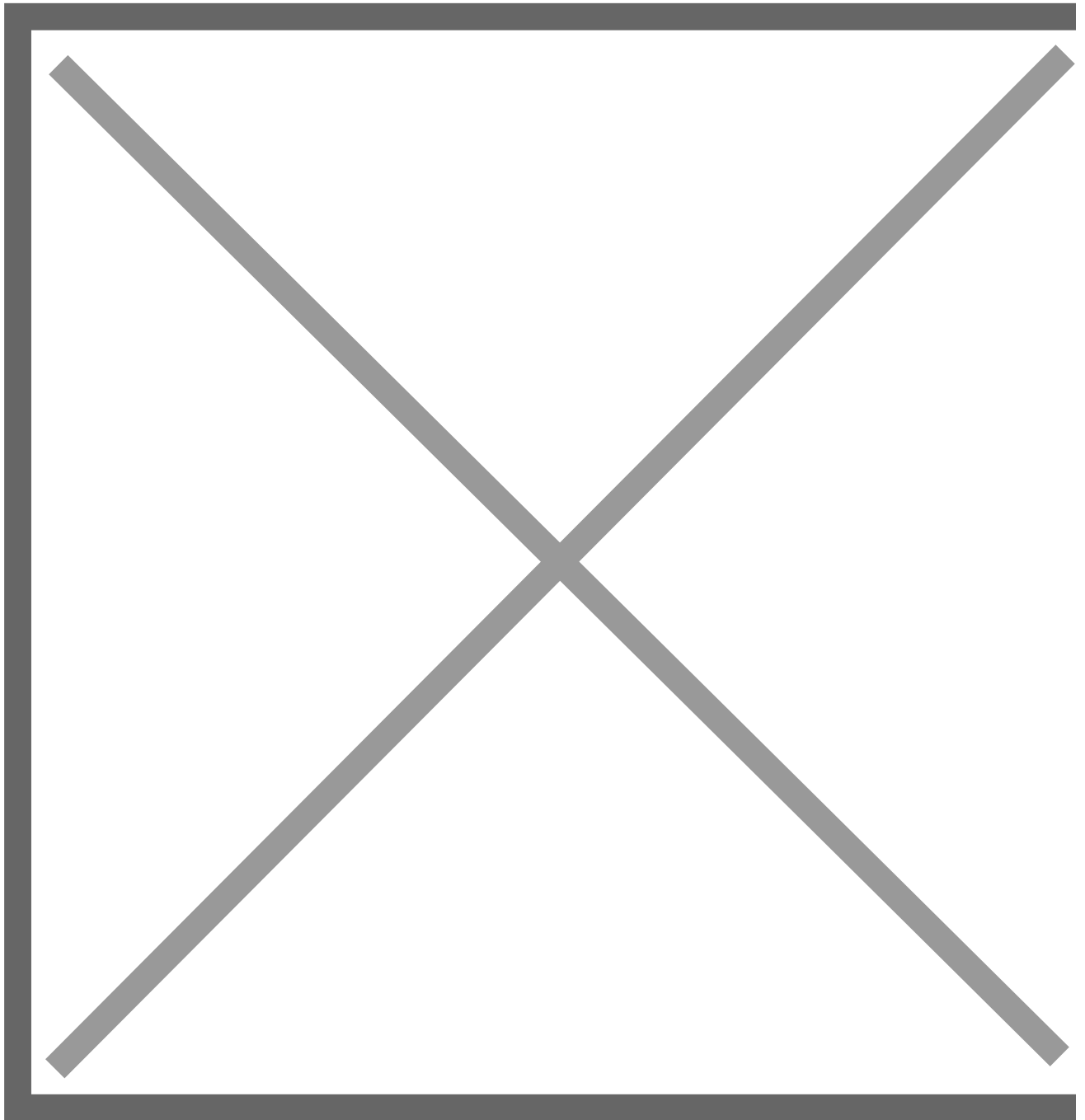


Vous pouvez déjà noter que contrairement à la commande native qui serait "**winget search "7-Zip"**", le résultat obtenu est un objet tableau et non du texte ! Ce qui signifie que vous pouvez stocker le résultat dans une variable PowerShell et le manipuler, ce qui offre beaucoup de possibilités.

Installer un paquet

Pour installer le paquet 7-Zip dans sa dernière version stable, exécutez cette commande :

```
Install-WinGetPackage -Id "7zip.7zip"
```



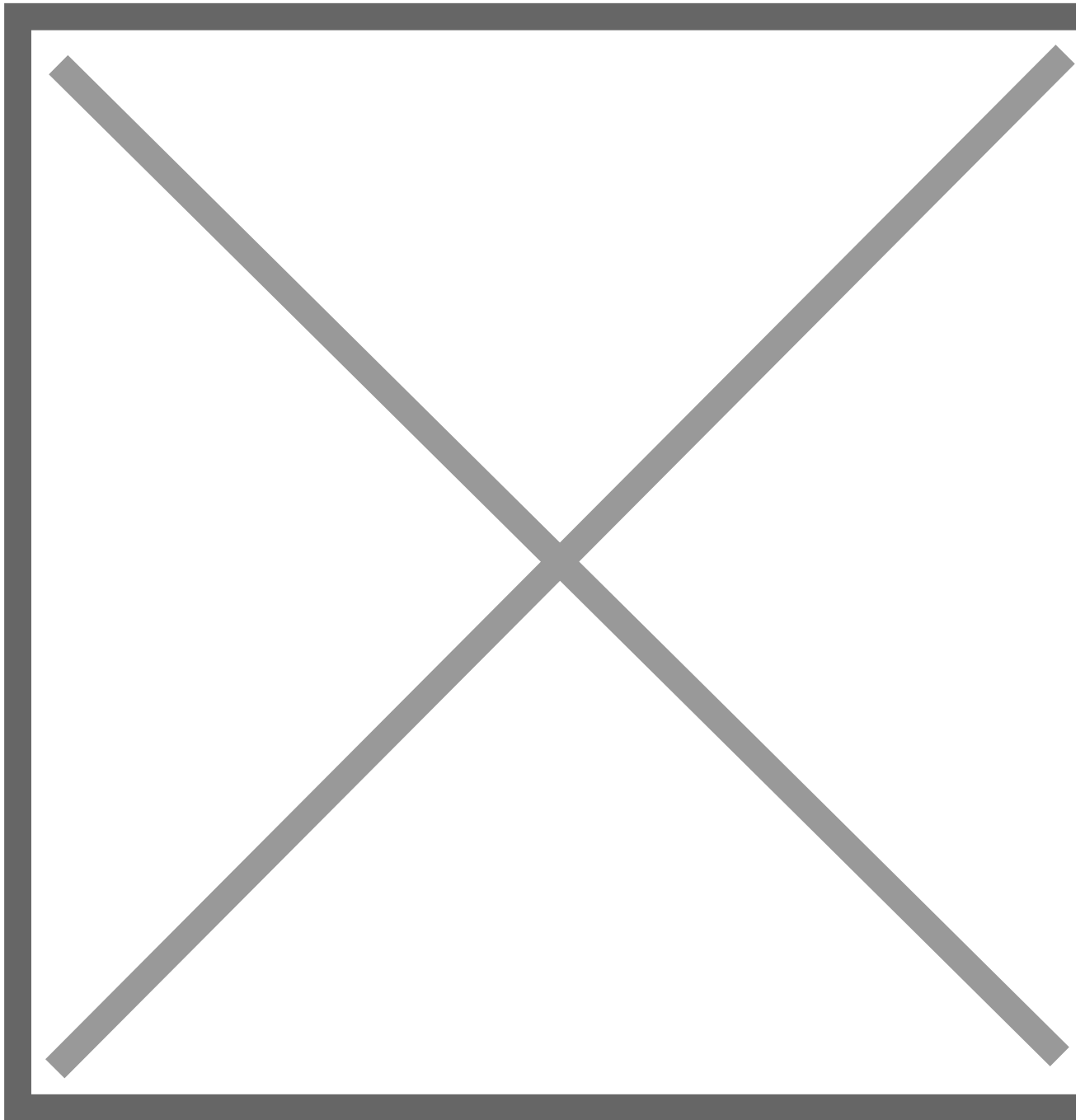
À noter qu'il est également possible de profiter de la commande précédemment utilisée et d'utiliser la pipeline :

```
Find-WinGetPackage -Id "7zip.7zip" | Install-WinGetPackage
```

Chose à savoir, la commande "**Install-WinGetPackage**" regorge de fonctionnalités grâce à ses paramètres, il est, par exemple, possible de tester l'installation avant même d'exécuter réellement celle-ci via le paramètre **-WhatIf** :

```
Install-WinGetPackage -Id "7zip.7zip" -WhatIf | Format-List
```

Ce qui donne :



Il est également possible de préciser une version (**-Version**), une architecture (**-Architecture**) x64 ou x86 par exemple, le mode (**-Mode**) tel qu'**Interactif** ou **Silent**... Je vous laisse découvrir la documentation grâce à la commande suivante :

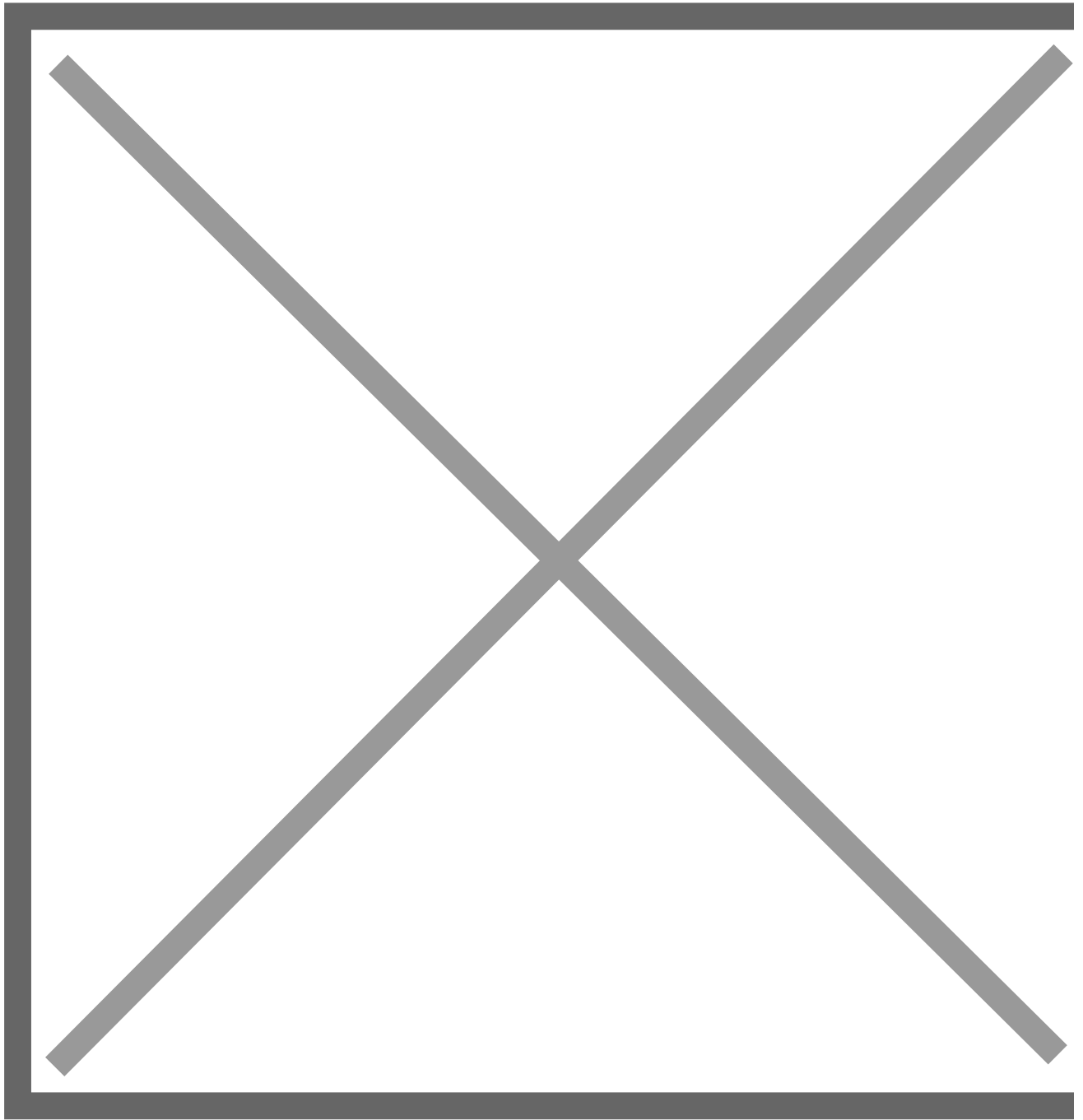
```
Get-Help Install-WinGetPackage -Full
```

Lister les paquets installés

Pour afficher la liste des paquets installés sur votre système via Winget, utilisez la commande suivante :

```
Get-WinGetPackage
```

Cette commande retourne dans la console une liste détaillée des logiciels installés, incluant leur nom, leur version, et leur ID.



Mettre à jour un paquet

Pour mettre à jour un paquet spécifique avec Winget, utilisez :

```
Update-WinGetPackage -Id "Id_du_paquet"
```

Désinstaller un paquet

Pour désinstaller un paquet spécifique, utilisez la commande suivante :

```
Uninstall-WinGetPackage -Id "Id_du_paquet"
```

Vous pouvez identifier l'ID d'un paquet en listant les paquets installés avec "**Get-WinGetPackage**".

Exporter la liste des paquets installés dans un fichier CSV

Pour sauvegarder la liste des paquets installés dans un fichier au format CSV, exécutez la commande ci-dessous :

```
Get-WinGetPackage | Export-Csv -Path "C:\exemple\paquets_list.csv"
```

Cet exemple va créer le fichier "**C:\exemple\paquets_list.csv**" en sortie.

Installation en masse de paquets à partir d'une liste

Vous pouvez aussi importer une liste de paquets à installer à partir d'un fichier CSV et procéder à l'installation en masse de toutes les applications référencées dans ce fichier CSV. Une boucle `ForEach` permet de traiter l'ensemble du fichier CSV :

```
$packages = Import-Csv -Path "C:\exemple\paquets_list.csv"
foreach ($package in $packages) {
    Install-WinGetPackage -Id $package.Id
}
```

Le cmdlet Repair-WinGetPackageManager

Cette commande est particulière, elle permet non seulement de réparer l'installation du client WinGet (ou de le mettre à jour), mais elle permet tout simplement aussi de l'installer ! Ce qui veut dire que vous pouvez d'abord installer le module, exécuter cette commande, puis utiliser WinGet directement.

```
Repair-WinGetPackageManager -Latest -Force
```

Conclusion

L'utilisation du module PowerShell **Microsoft.WinGet.Client** présente plusieurs avantages par rapport à l'utilisation classique de WinGet via la ligne de commande :

- **Intégration avec PowerShell** : les commandes retournent des objets PowerShell, ce qui permet de les manipuler facilement avec d'autres commandes PowerShell.
- **Simplification des scripts** : les commandes PowerShell sont souvent plus concises et plus faciles à lire que les commandes en ligne de commande.

Le module simplifie grandement l'utilisation de WinGet en intégrant des commandes natives PowerShell. Que vous soyez un débutant ou un utilisateur avancé, ce module vous permettra de gérer vos applications de manière plus efficace et plus intuitive.