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**.

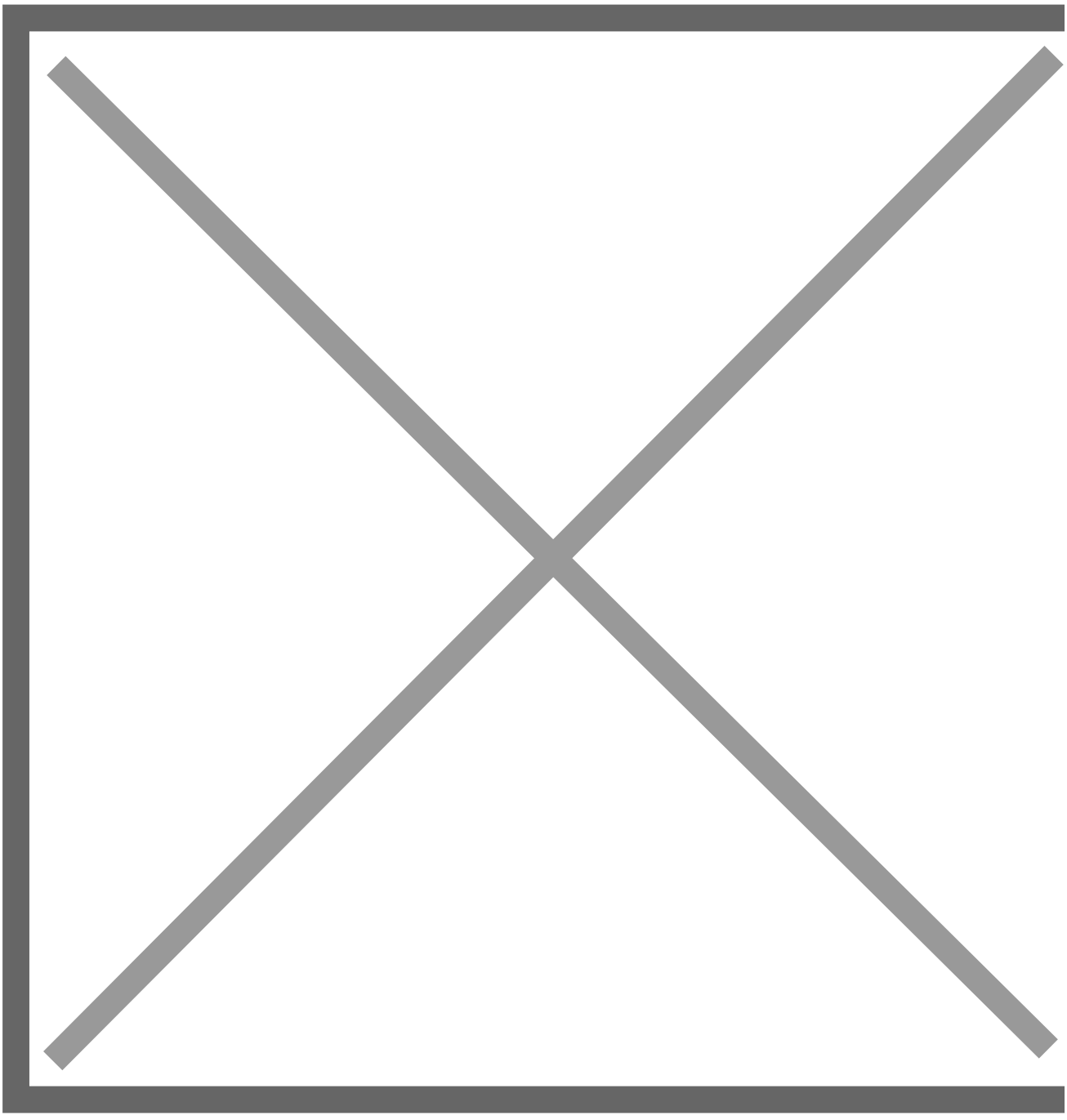
Ensuite, importez le module :

```
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

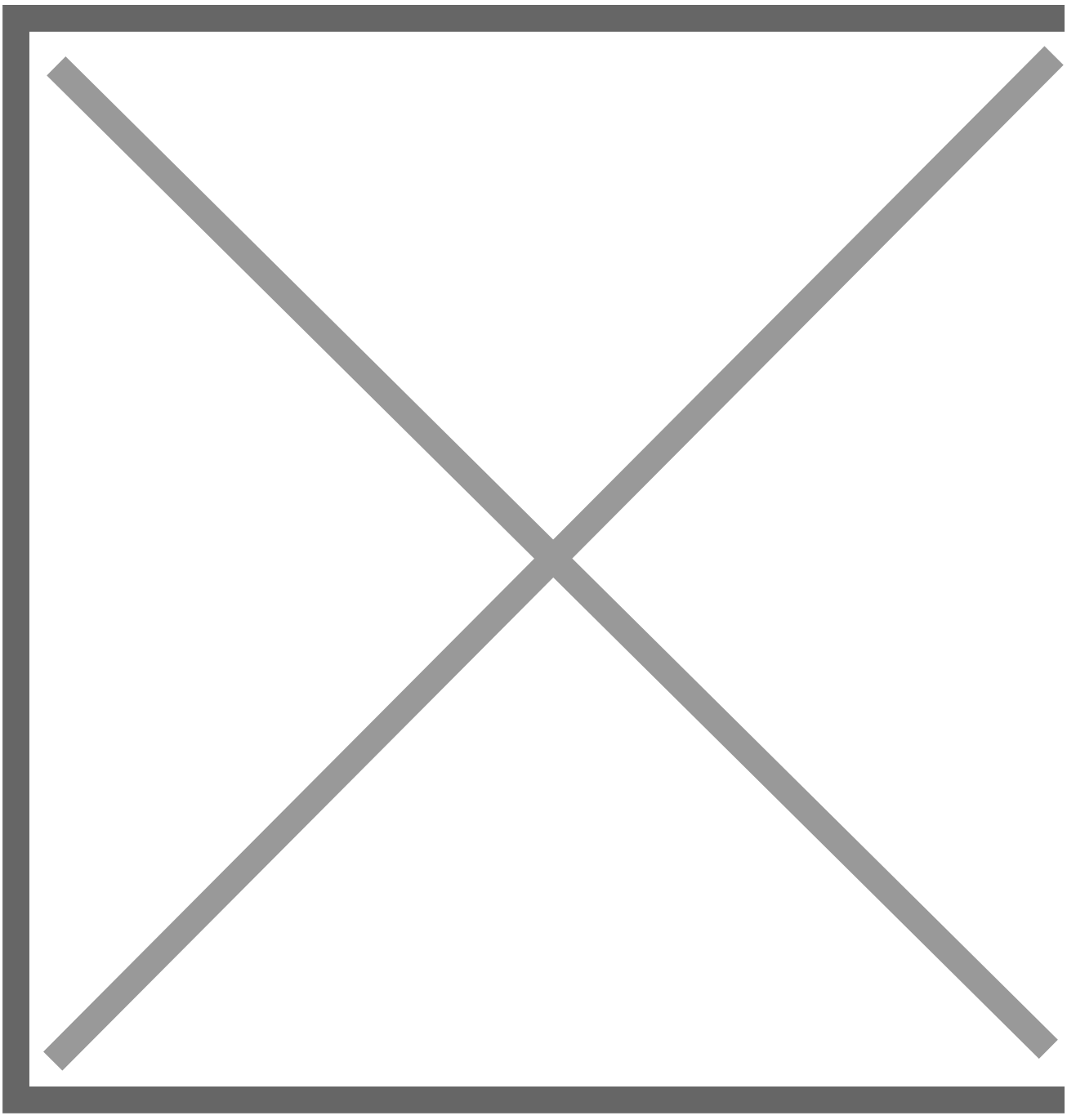
Une fois le module installé et importé, vous pouvez commencer à utiliser les commandes natives PowerShell pour gérer vos applications.

Rechercher un paquet

(exemple : 7-Zip)

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

f

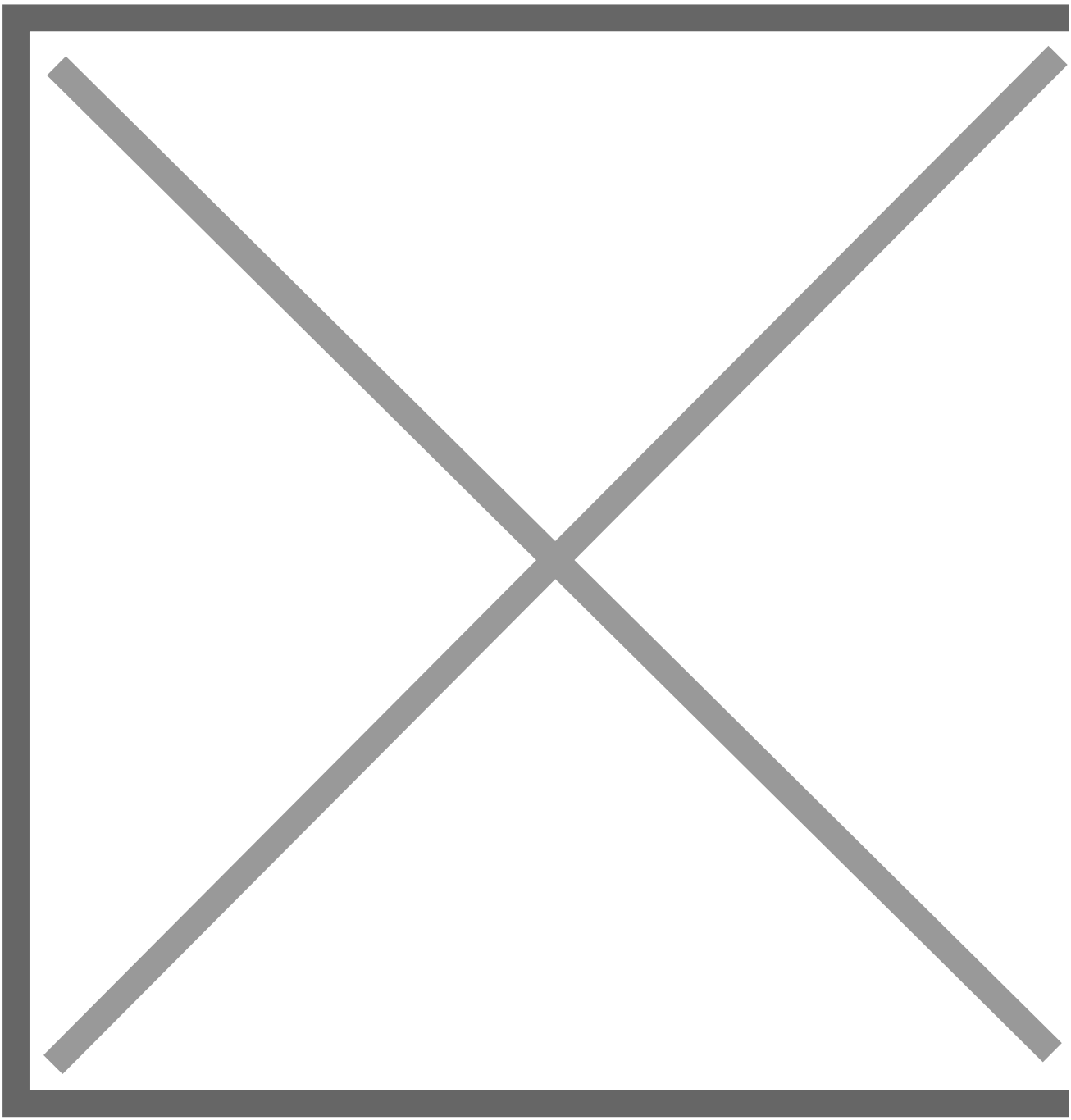


Vous pouvez déjà noter que contrairement à la commande native qui serait "**winget search "7-Zip"**", le résultat obtenu est un objet tableau et non du texte ! Ce qui signifie que vous pouvez stocker le résultat dans une variable PowerShell et le manipuler, ce qui offre beaucoup de possibilités.

Installer un paquet

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

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



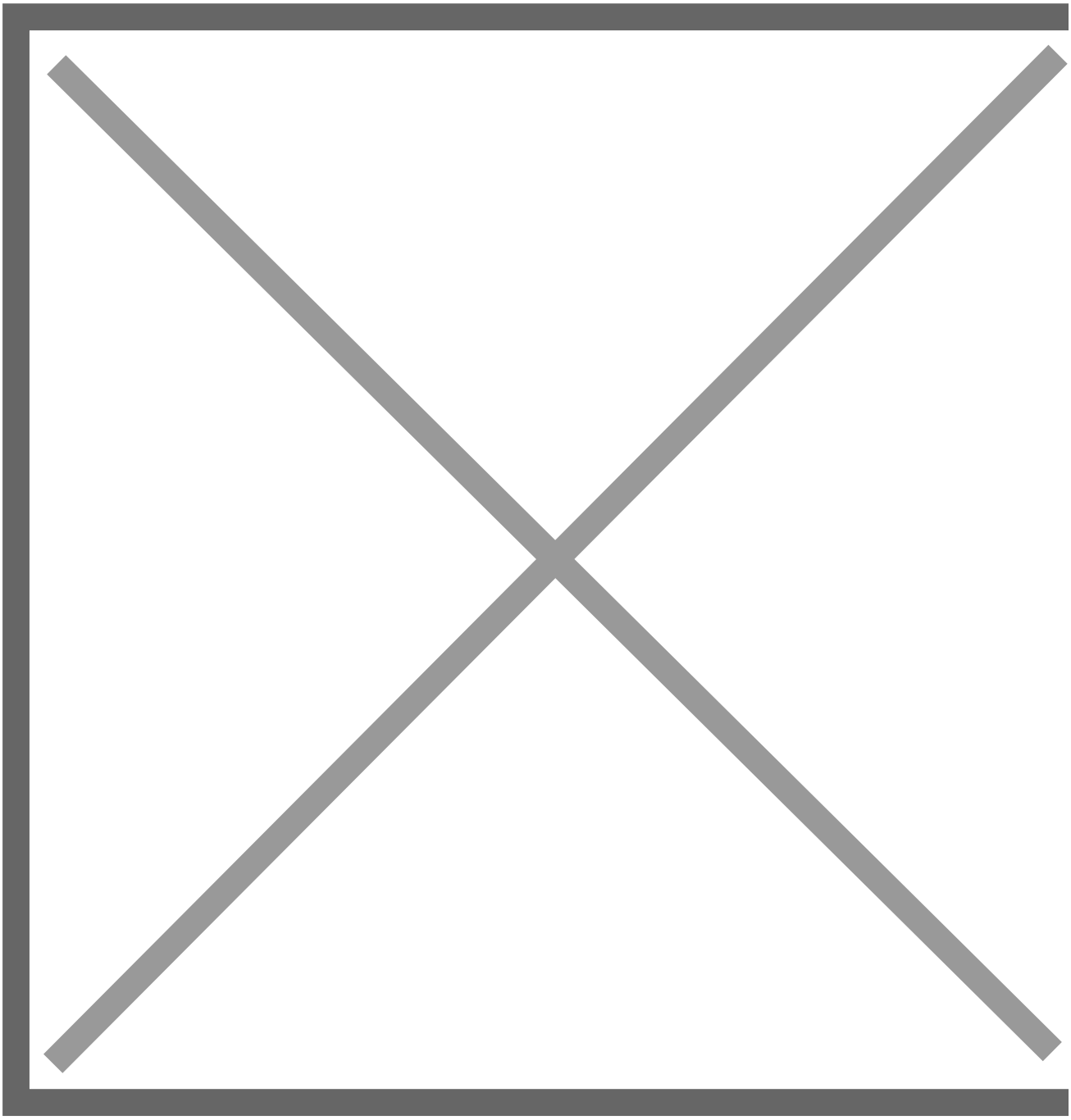
À noter qu'il est également possible de profiter de la commande précédemment utilisée et d'utiliser la pipeline :

```
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 (**-Architechture**) x64 ou x86 par exemple, le mode (**-Mode**) tel qu'**Interactif** ou **Silent**... Je vous laisse découvrir la documentation grâce à la commande suivante :

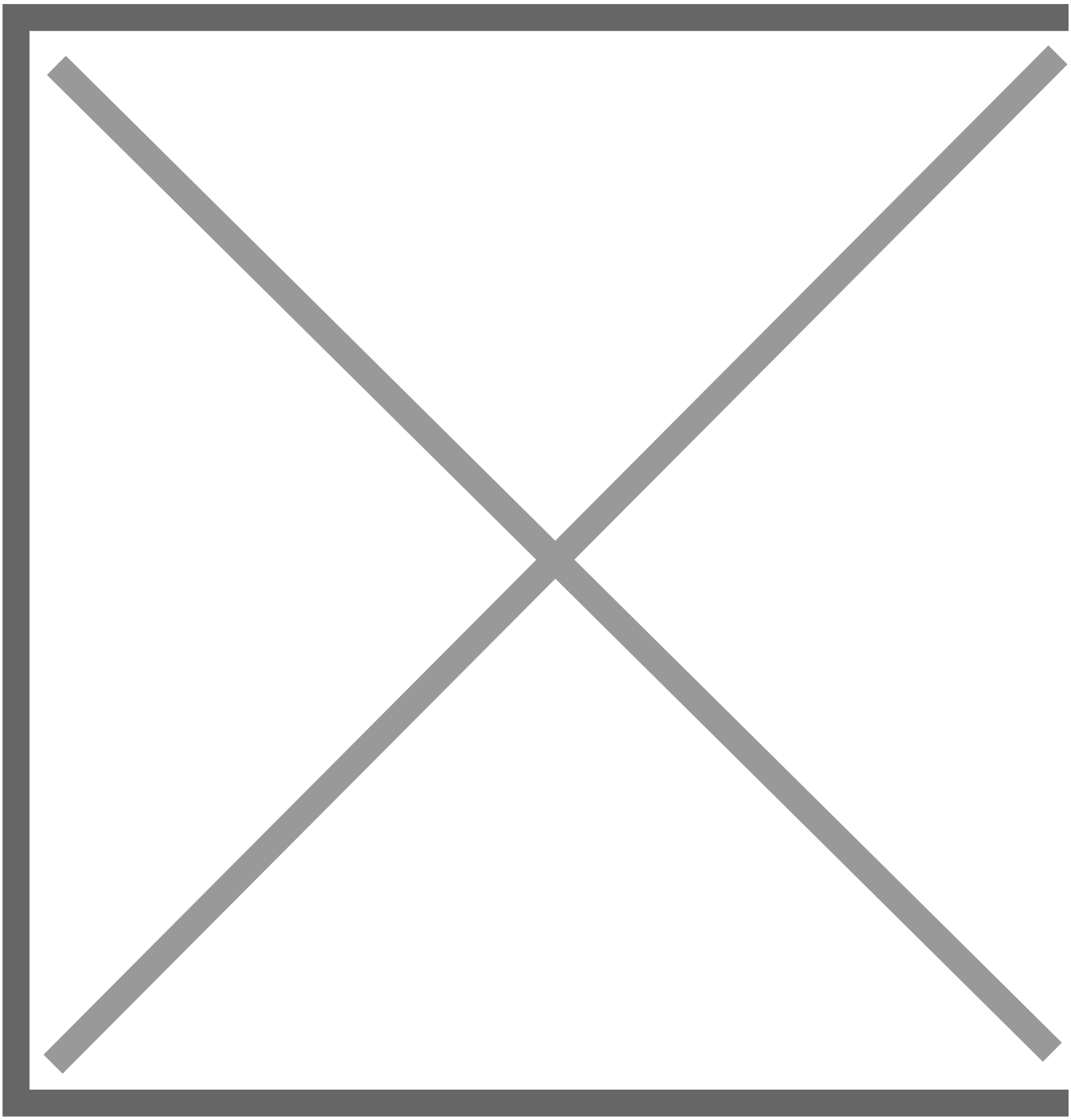
```
Get-Help Install-WinGetPackage -Full
```

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.

Révision #1
Créé 12 janvier 2025 14:44:22 par Angelo
Mis à jour 1 avril 2025 18:43:52 par Angelo