Types d'objets de base

History

D'abord et avant tout, je vous suggère fortement de vous familiariser avec le type d'objet History. Il s'agit de l'historique des commandes que vous avez saisi au cours d'une session Powershell. Ces objets vous aideront à créer des scritps, à réviser vos commandes, etc.

Ces commandes sont écrites dans le fichier "%AppData%\Microsoft\Windows\PowerShell\PSReadLine"

Ces objets sont consultables à l'aide de la commande "Get-History" et sont composés des propriétés suivantes :

```
Id                : 7
CommandLine       : $mon_message = "Hello world!"
ExecutionStatus   : Completed
StartExecutionTime : 2025-01-18 4:20:04 PM
EndExecutionTime   : 2025-01-18 4:20:04 PM

Id                : 8
CommandLine       : Write-Host $mon_message
ExecutionStatus   : Completed
StartExecutionTime : 2025-01-18 4:20:10 PM
EndExecutionTime   : 2025-01-18 4:20:10 PM
```

Vous pourriez donc facilement à partir de cet historique créer un script basé sur les manipulations que vous avez fait lors de votre session si vous sélectionnez seulement la ligne de commande comme ceci :

```
PS C:\Users\alexa> Get-History | Select-Object -Property CommandLine


CommandLine
-----------
clear-history
$mon_message = "Hello world!"
Write-Host $mon_message
Get-History
```

```
Get-History | Select-Object -Property *
```

ID

L'identifiant de l'objet détermine l'ordre dans lequel les commandes ont été exécutées.

CommandLine

La commande exécutée.

ExecutionStatus

L'état de la commande à savoir si elle est en cours d'exécution (ex. un ping continu), si elle a échoué ou si elle a été complétée.

StartExecutionTime

La date et l'heure de l'exécution de la commande.

EndExecutionTime

La date et l'heure de la fin d'exécution de la commande, qu'elle ait été annulée, qu'elle ait échouée ou qu'elle se soit complétée.

Item et ChildItem

Un ChildItem peut représenter plusieurs types d'information. Ce type d'objet est généralement utilisé pour manipuler des fichiers sur un espace de stockage mais peut aussi être utilisé pour modifier des clés de registre.

On peut déduire du nom des types d'objets qu'un Item sélectionne seulement l'objet demandé tandis qu'un ChildItem sélectionnera les objets situés à l'intérieur de l'objet. Ces deux types d'objets possèdent les mêmes propriétés et peuvent être manipulés de la même façon.

```
PS C:\Users\alexa> Get-Item D:\Item | Select-Object -Property *



PSPath            : Microsoft.PowerShell.Core\FileSystem::D:\Item
PSParentPath      : Microsoft.PowerShell.Core\FileSystem::D:\
PSChildName       : Item
PSDrive           : D
PSProvider        : Microsoft.PowerShell.Core\FileSystem
PSIsContainer     : True
Mode              : d-----
BaseName          : Item
```

```
Target           : {}
LinkType         :
Name             : Item
FullName         : D:\Item
Parent           :
Exists           : True
Root             : D:\
Extension        :
CreationTime     : 2025-01-18 4:35:18 PM
CreationTimeUtc  : 2025-01-18 9:35:18 PM
LastAccessTime   : 2025-01-18 4:35:27 PM
LastAccessTimeUtc : 2025-01-18 9:35:27 PM
LastWriteTime    : 2025-01-18 4:35:26 PM
LastWriteTimeUtc  : 2025-01-18 9:35:26 PM
Attributes       : Directory
```

Lorsque l'on récupère les "ChildItem", il est aussi possible d'ajouter le paramètre "-recurse" pour sélectionner récursivement les objets dans les sous-dossiers ou sous-clés de registre.

```
PS C:\Users\alexa> Get-ChildItem D:\Item


    Directory: D:\Item


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        2025-01-18   4:40 PM                Subdirectory
-a----        2025-01-18   4:35 PM              0 ChildItem-1.txt
-a----        2025-01-18   4:38 PM              0 ChildItem-2.txt



PS C:\Users\alexa> Get-ChildItem D:\Item -Recurse


    Directory: D:\Item


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        2025-01-18   4:40 PM                Subdirectory
-a----        2025-01-18   4:35 PM              0 ChildItem-1.txt
-a----        2025-01-18   4:38 PM              0 ChildItem-2.txt
```

```
    Directory: D:\Item\Subdirectory


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        2025-01-18   4:40 PM              0 SubChildItem.txt
```

L'exemple suivant servirait à consulter le registre HKEY_CURRENT_USER.

```
PS C:\Users\alexa> Get-ChildItem HKCU:/Item


    Hive: HKEY_CURRENT_USER\Item

Name                           Property
----                           --------
ChildItem                      Value : Content
```

Content

Tout comme en Linux la commande "cat" nous permet de lire le contenu d'un fichier et la redirection de sortie > et >> nous permet d'ajouter ou modifier le contenu du fichier, le type d'objet "Content" en Powershell nous permet de poser des actions similaires. Ce sont ici les commandes Get-Content, Clear-Content, Add-Content et Set-Content qui nous intéressent.

```
# Contenu initial du fichier
PS C:\Users\alexa> Get-Content D:\Item\ChildItem-1.txt
Hello world!

# Contenu vidé du fichier
PS C:\Users\alexa> Clear-Content D:\Item\ChildItem-1.txt
PS C:\Users\alexa> Get-Content D:\Item\ChildItem-1.txt

# Contenu ajouté au fichier
PS C:\Users\alexa> Add-Content -Path D:\Item\ChildItem-1.txt -Value "Bonjour le monde!"
PS C:\Users\alexa> Get-Content D:\Item\ChildItem-1.txt
Bonjour le monde!

# Contenu du fichier remplacé
PS C:\Users\alexa> Set-Content -Path D:\Item\ChildItem-1.txt -Value "Goodbye world!"
PS C:\Users\alexa> Get-Content D:\Item\ChildItem-1.txt
Goodbye world!
```

ACL

Les ACL ou "Access Control List" sont les permissions d'accès assignée à différents fichiers. Ces listes peuvent être consultées ou modifiées en Powershell à l'aide des commandes "Get-ACL" et "Set-ACL". Contrairement à "chmod" sous Linux nous permettant seulement de modifier les autorisations du fichier pour son propriétaire, son groupe propriétaire et les autres, la manipulation des ACL avec Powershell nous permet de gérer les ACLs avancées (ex. groupes et utilisateurs supplémentaires) tout comme "setfact" et "getfacl" sous Linux.

Les propriétés les plus intéressantes de ce type d'objet sont le propriétaire, le groupe propriétaire et la propriété "AccessToString" déterminant quel utilisateur a accès au fichier.

```
# Vérification initiale des ACL (Les propriétés non pertinentes ont été retirées de
l'affichage)
PS C:\Users\alexa> Get-ACL D:\Item\ChildItem-1.txt | Select-Object -Property *

Path                   : Microsoft.PowerShell.Core\FileSystem::D:\Item\ChildItem-1.txt
Owner                  : PC\alexa
Group                  : PC\alexa
AccessToString         : BUILTIN\Administrators Allow  FullControl
                         NT AUTHORITY\SYSTEM Allow  FullControl
                         NT AUTHORITY\Authenticated Users Allow  Modify, Synchronize
                         BUILTIN\Users Allow  ReadAndExecute, Synchronize

# Ajout d'une permission
#
# Récupérer les permissions actuelles
PS C:\Users\alexa> $acl = Get-ACL D:\Item\ChildItem-1.txt
# Créer une nouvelle permission et la traduire en objet de règle
PS C:\Users\alexa> $permission = "PC\Invité","Modify","None",'None','Allow'
PS C:\Users\alexa> $rule = New-Object System.Security.AccessControl.FileSystemAccessRule
$permission
# Ajouter la règle à l'objet de permissions
PS C:\Users\alexa> $acl.AddAccessRule($rule)
# Appliquer le nouvel objet de permissions au fichier
PS C:\Users\alexa> Set-ACL -Path D:\Item\ChildItem-1.txt -AclObject $acl

# Valider les nouvelles permissions, observer que "Invité" a été ajouté aux permissions (Les
propriétés non pertinentes ont été retirées de l'affichage)
PS C:\Users\alexa> Get-ACL -Path D:\Item\ChildItem-1.txt | Select-Object -Property *
```

```
Path                   : Microsoft.PowerShell.Core\FileSystem::D:\Item\ChildItem-1.txt
Owner                  : PC\alexa
Group                  : PC\alexa
AccessToString         : PC\Invité Allow  Modify, Synchronize
                         BUILTIN\Administrators Allow  FullControl
                         NT AUTHORITY\SYSTEM Allow  FullControl
                         NT AUTHORITY\Authenticated Users Allow  Modify, Synchronize
                         BUILTIN\Users Allow  ReadAndExecute, Synchronize
```

Comme on peut l'observer ici, il arrive souvent que les objets deviennent rapidement complexes à créer et intégrer.

Service

Un autre objet pertinent à savoir manipuler dans un contexte d'automatisation est un service. À l'aide des commandes Start, Stop, Restart et Set, vous pourrez manipuler les différents services en exécution sur votre machine. Il est important pour ce type d'objet de faire la différence entre le nom du service et son nom d'affichage pour bien manipuler ces objets.

```
PS C:\Users\alexa> Get-Service | Where-Object Name -like wuauserv | Select-Object -Property *


Name                : wuauserv
RequiredServices    : {rpcss}
CanPauseAndContinue : False
CanShutdown         : False
CanStop             : False
DisplayName         : Windows Update
DependentServices   : {}
MachineName         : .
ServiceName         : wuauserv
ServicesDependedOn  : {rpcss}
ServiceHandle       :
Status              : Stopped
ServiceType         : Win32OwnProcess, Win32ShareProcess
StartType           : Manual
Site                :
Container           :
```

On peut observer ici le service de Windows Update et son type de démarrage qui est manuel. Ce service démarre seulement lorsqu'une recherche ou installation de mises à jour soit déclenchée, que ce soit manuellement ou par tâche planifiée.

Revision #10
Created 2025-01-18 21:06:39 UTC by Alexandre Arsenault-Jetté
Updated 2025-01-21 03:40:54 UTC by Alexandre Arsenault-Jetté