MODULE 0

Introduction à Powershell

Objectifs de ce module :

- ✓ ***Connaître les éléments de base de Powershell***
- ✓ ***Installer et utiliser la console powershell***
- ✓ ***Utiliser un environnement graphique pour Powershell***

Table des matières

Sujets ***Page***

Introduction

Windows PowerShell™ est un nouvel environnement de ligne de commande spécialement conçu pour les administrateurs système. Il comprend une invite interactive et un environnement de script qui peuvent être utilisés indépendamment l'un de l'autre ou ensemble.

Windows PowerShell est conçu pour améliorer l'environnement de ligne de commande et de script en éliminant des problèmes de longue date et en ajoutant de nouvelles fonctionnalités.

Contrairement à la plupart des interpréteurs de commandes qui acceptent et retournent du texte, Windows PowerShell est basé sur le Common Language Runtime (CLR) .NET Framework et accepte et retourne des objets .NET Framework. Cette modification fondamentale de l'environnement apporte des outils et méthodes entièrement nouveaux pour la gestion et la configuration de Windows.

Windows PowerShell introduit le concept « d'applet de commande », un outil de ligne de commande simple, à fonction unique, intégré dans l'interpréteur de commandes. Vous pouvez utiliser chaque applet de commande séparément, mais leur puissance se révèle lorsque vous combinez ces outils simples pour effectuer des tâches complexes. Windows PowerShell inclut plus de cent applets de commande principales. Vous pouvez également écrire vos propres applets de commande et les partager avec d'autres utilisateurs.

Comme de nombreux interpréteurs de commandes, Windows PowerShell vous donne accès au système de fichiers de l'ordinateur. En outre, les *fournisseurs* Windows PowerShell vous permettent d'accéder à d'autres magasins de données, tels que le Registre et les magasins de certificats de signatures numériques, aussi facilement que vous accédez au système de fichiers.

Concepts importants

La conception de Windows PowerShell intègre des concepts provenant de nombreux environnements différents. En fait, Powershell assimile plusieurs éléments qui proviennent de d'autres interpréteurs de commandes connus ou de d'autres éléments de langages.

Les commandes ne sont pas basées sur du texte

Contrairement aux commandes d'interface de ligne de commande traditionnelles, les applets de commande Windows PowerShell sont conçues pour traiter des objets, informations structurées allant au-delà de simples chaînes de caractères apparaissant à l'écran. La sortie d'une commande contient toujours des informations supplémentaires que vous pouvez utiliser si vous en avez besoin.

Windows PowerShell gère l'entrée et l'affichage sur une console

Les interpréteurs de commandes traditionnels ont leurs propres façon d'afficher l'aide sur une commande. Certains utilisent « -? » d'autres « /H » et même le fameux « /? ». Powershell rend uniforme l'aide en utilisant la même applet de commande et ce pour toute les commandes pour lesquelles vous voulez obtenir l'aide.

Il est important de comprendre que les fonctions d'aide sont disponibles dans Windows PowerShell même lorsque vous y exécutez des outils de ligne de commande traditionnels. Windows PowerShell traite les paramètres et passe les résultats aux outils externes.

La famille de commandes est extensible

Les interfaces telles que Cmd.exe ne vous permettent pas d'étendre directement le jeu de commandes intégré. Vous pouvez créer des outils de ligne de commande externes qui s'exécutent dans Cmd.exe, mais ces outils sont dépourvus de services, tels que l'intégration de l'Aide, et Cmd.exe ne peut pas déterminer automatiquement qu'il s'agit de commandes valides.

Les commandes binaires natives de Windows PowerShell, appelées « *applets de commande* », peuvent être enrichies des applets de commande que vous créez et ajoutez à Windows PowerShell. Vous pouvez ainsi enrichir l'interpréteur de commande avec vos nouveaux applets de commande personnel.

Windows PowerShell utilise une syntaxe C#
Parce qu'il se base sur le .NET Framework, Windows PowerShell présente des fonctionnalités de syntaxe et des mots clés très semblables à ceux utilisés en langage de programmation C#. Comme la sortie d'une commande est un objet, vous retrouverez certaines propriétés propres aux objets qui sont disponibles dans le langage C#.

Éléments de base de powershell

Applets de commande « verbe-nom »

Powershell permet de reconnaître ou d'apprendre les commandes de façon plus simple. En fait, il s'agit de connaître un groupe restreint de commandes pour pouvoir réaliser un bon nombre d'opérations communes.

> Les applets de commande utilisent un format « verbe-nom » pour réduire la mémorisation des commandes

Windows PowerShell utilise un système de dénomination « verbe-nom », où chaque nom d'applet de commande est constitué d'un verbe standard et d'un nom spécifique, tous deux souvent anglais, séparés par un tiret. Les verbes Windows PowerShell ne sont pas toujours des verbes anglais, mais ils expriment des actions spécifiques dans Windows PowerShell. Les noms sont très similaires à ceux des autres langues et décrivent des types spécifiques d'objets essentiels pour l'administration du système. Il est aisé de démontrer par quelques exemples comment ces noms en deux parties réduisent l'effort d'apprentissage.

Les noms sont moins restreints, mais doivent toujours décrire ce sur quoi agit une commande. Windows PowerShell a des commandes telles que **Get-Process**, **Stop-Process**, **Get-Service** et **Stop-Service**.

Si vous considérez un jeu standard de 10 verbes et 10 noms, vous n'avez que 20 mots à retenir, mais ces mots peuvent être utilisés pour former 100 commandes distinctes.

Vous pouvez fréquemment reconnaître ce que fait une commande en lisant son nom . Par exemple, une commande d'arrêt de l'ordinateur peut être **Stop-Computer**. Une commande qui répertorie tous les ordinateurs d'un réseau peut être **Get-Computer**. La commande qui obtient la date système est **Get-Date**.

Vous pouvez répertorier toutes les commandes qui incluent un verbe particulier au moyen du paramètre **-Verb** de **Get-Command** (nous présenterons en détail **Get-Command** dans la section suivante). Par exemple, pour consulter toutes les applets de commande qui utilisent le verbe **Get**, tapez :

```
PS> Get-Command -Verb Get
CommandType     Name                              Definition
-----------     ----                              ----------
Cmdlet          Get-Acl                           Get-Acl [[-Path]
<String[]>]...
Cmdlet          Get-Alias                         Get-Alias [[-Name]
<String[]...
Cmdlet          Get-AuthenticodeSignature         Get-AuthenticodeSignature
[-...
Cmdlet          Get-ChildItem                     Get-ChildItem [[-Path]
<Stri...
...
```

Le paramètre **-Noun** est plus utile encore, car il vous permet de consulter une famille des commandes qui affectent le même type d'objet. Par exemple, si vous souhaitez consulter les commandes disponibles pour la gestion de services, tapez la commande suivante :

```
PS> Get-Command -Noun Service
CommandType     Name                              Definition
-----------     ----                              ----------
Cmdlet          Get-Service                       Get-Service [[-Name]
<String...
Cmdlet          New-Service                       New-Service [-Name]
<String>...
Cmdlet          Restart-Service                   Restart-Service [-Name]
<Str...
Cmdlet          Resume-Service                    Resume-Service [-Name]
<Stri...
Cmdlet          Set-Service                       Set-Service [-Name]
<String>...
Cmdlet          Start-Service                     Start-Service [-Name]
<Strin...
Cmdlet          Stop-Service                      Stop-Service [-Name]
<String...
Cmdlet          Suspend-Service                   Suspend-Service [-Name]
<Str...
...
```

Une commande n'est pas une applet de commande simplement parce que son nom obéit au modèle verbe-substantif. Clear-Host, par exemple, est une commande Windows PowerShell native qui permet d'effacer le contenu de la fenêtre de console, mais n'est pas une applet de commande. La commande Clear-Host est en réalité une fonction interne, comme vous pouvez le voir si vous exécutez Get-Command dessus :

```
PS> Get-Command -Name Clear-Host

CommandType     Name                               Definition
-----------     ----                               ----------
Function        Clear-Host                         $spaceType =
[System.Managem...
```

Paramètres des applets de commandes

Paramètre « -? »

Lorsque vous spécifiez le paramètre **-?** pour une applet de commande, celle-ci n'est pas exécutée. À la place, Windows PowerShell affiche l'aide qui lui est associée.

Paramètres courants

Windows PowerShell propose plusieurs paramètres appelés « *paramètres courants* ». Parce que ces paramètres sont contrôlés par le moteur Windows PowerShell, chaque fois qu'ils sont implémentés par une applet de commande, ils se comportent toujours de la même manière. Les paramètres courants sont **WhatIf**, **Confirm**, **Verbose**, **Debug**, **Warn**, **ErrorAction**, **ErrorVariable**, **OutVariable** et **OutBuffer**.

Paramètres suggérés

Les applets de commande principales Windows PowerShell utilisent des noms standard pour les paramètres semblables. Bien que l'utilisation de noms de paramètres ne soit pas imposée, des indications explicites sont fournies afin de favoriser la normalisation.

Par exemple, ces indications recommandent de nommer un paramètre faisant référence à un ordinateur par son nom comme **ComputerName**, plutôt que par Server, Host, System, Node ou autres mots courants possibles. Parmi les noms de paramètre suggérés importants figurent **Force**, **Exclude**, **Include**, **PassThru**, **Path** et **CaseSensitive**.

Aide sur les commandes

Windows PowerShell inclut des rubriques d'aide détaillées qui expliquent les concepts et le langage Windows PowerShell. Il existe également des rubriques d'aide pour chaque applet de commande et fournisseur, ainsi que des rubriques d'aide pour de nombreux scripts et fonctions.

Vous pouvez afficher ces rubriques d'aide à l'invite de commandes ou afficher les versions mises à jour les plus récentes de ces rubriques dans la bibliothèque TechNet de Microsoft. De nombreux programmes qui hébergent Windows PowerShell, tels que l'environnement d'écriture de scripts intégré Windows PowerShell Integrated Scripting Environment, proposent des fonctions d'aide supplémentaires, telles que l'aide contextuelle et un fichier d'aide compilé (.chm).

L'applet de commande pour afficher l'aide est : **get-help**

Par exemple, pour afficher l'aide sur la commande get-command, on peut faire :

get-help get-command ou encore

help get-command

get-command -?

man get-command (hey oui, vous retrouverez ici une commande connue sous Linux)

Plus de détails dans le module portant sur l'aide des applets de commande.

Utilisation de noms de commandes familiers

Windows PowerShell permet aux utilisateurs de faire référence aux commandes par d'autres noms, appelés *alias*. Grâce à ces alias, les utilisateurs ayant l'expérience d'autres interpréteurs de commandes peuvent réutiliser les noms de commandes qu'ils connaissent pour effectuer des opérations similaires dans Windows PowerShell. Si nous n'abordons pas les alias dans le détail, vous pouvez toujours les utiliser pour faire vos premiers pas dans Windows PowerShell.

Un alias vous permet d'associer un nom de commande que vous tapez à une autre commande. Par exemple, Windows PowerShell a une fonction interne nommée **Clear-Host** qui efface le contenu de la fenêtre de sortie. Si vous tapez la commande **cls** ou **clear** à une invite de commandes, Windows PowerShell l'interprète en tant qu'alias de la fonction **Clear-Host** et exécute la fonction **Clear-Host**.

Si vous avez utilisé Cmd.exe pendant des années, lorsque vous obtenez un écran complet de sortie et souhaitez le nettoyer, vous pouvez par réflexe taper la commande **cls** et appuyer sur la touche Entrée. Sans l'alias de la fonction **Clear-Host** dans Windows PowerShell, vous obtiendriez simplement le message d'erreur « **'cls' is not recognized as a cmdlet, function, operable program, or script file.** » et n'auriez aucune idée de la marche à suivre pour effacer la sortie.

Vous trouverez ci-dessous une courte liste des commandes Cmd.exe et UNIX courantes que vous pouvez utiliser dans Windows PowerShell.

cat	dir	mount	rm
cd	echo	move	rmdir
chdir	erase	popd	sleep
clear	h	ps	sort
cls	history	pushd	tee
copy	kill	pwd	type
del	lp	r	write
diff	ls	ren	

Si, par réflexe, vous utilisez l'une de ces commandes et souhaitez apprendre le véritable nom de la commande Windows PowerShell native, vous pouvez utiliser la commande **Get-Alias** :

```
PS> Get-Alias cls

CommandType     Name                                Definition
-----------     ----                                ----------
Alias           cls                                 Clear-Host
```

Interprétation des alias standard

Contrairement aux alias décrits plus haut, lesquels ont été conçus pour la compatibilité des noms avec d'autres interfaces, les alias intégrés de Windows PowerShell sont généralement conçus dans l'optique de la concision. Ces noms plus courts peuvent être tapés rapidement, mais sont impossibles à lire si vous ne savez pas à quoi ils font référence.

Alias les plus courants :

get-command	**gcm**
get-item	**gi**
set-item	**si**
get-location	**gl**
set-location	**sl**

Habituellement, l'alias est formé de la première lettre du verbe suivie de la première lettre du nom.

Utilisation de la complétion par tabulation

Tout comme dans Linux, il est possible de compléter une commande ou une fonction en tapant le début de la commande et taper la touche « tab » pour compléter le texte.

Par exemple : en tapant get-c suivi de « tab », le nom de la commande se complétera d'elle-même.

Installation de powershell

Windows powershell version 2 est disponible pour toutes les versions de Windows depuis Windows XP. Ceci inclus Windows server 2003/2008, Windows 2008 R2, Windows vista et Windows 7. Powershell est déjà préinstallé sur Windows 2008 R2 et sur Windows 7. Donc, si vous avez Windows 7, rien à faire, powershell est déjà intégré. Cependant, pour toute autre version précédent Windows 7 ou Windows 2008 R2, powershell doit être installé manuellement. De plus, nous devons manuellement installer powershell v4 ainsi que l'architecture .NET 4 pour Windows 7.

Depuis la sortie de Windows 10, Powershell est intégré directement dans le système. Windows 10 utilise maintenant la version 4 de Powershell.

Vous pouvez visiter le site de Microsoft à l'adresse « http://download.microsoft.com » et entrer « powershell 4 » dans la boîte de recherche.

Voici un extrait de ce qu'on y dit sur le site de Microsoft à propos de la version 4 :

WMF 4.0 ne peut être installé que sur les systèmes d'exploitation suivants.

- Windows 7 avec Service Pack 1
- Windows Server 2008 R2 avec Service Pack 1
- Windows Server 2012

Vous ne pouvez pas installer ce logiciel sur les ordinateurs exécutant Windows 8. Windows PowerShell 4.0 et d'autres fonctionnalités de ce package téléchargé sont disponibles en tant qu'éléments de Windows 8.1. La mise à niveau depuis une version complète de Windows 8 vers Windows 8.1 est gratuite ; pour en savoir plus sur cette mise à niveau, voir Mise à jour vers Windows 8.1.

WMF 4.0 nécessite Microsoft .NET Framework 4.5. Vous pouvez installer Microsoft .NET Framework 4.5 depuis le centre de téléchargement Microsoft.

Étape d'installation de Powerhell v4 sur Windows 7 :

1. Téléchargez .NET Framework 4.5 :

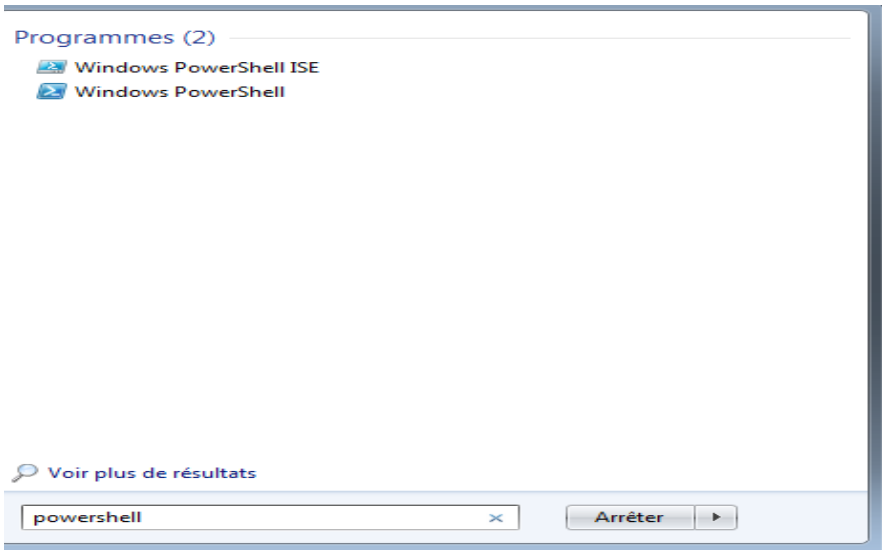
   http://www.microsoft.com/en-us/download/details.aspx?id=30653

2. Téléchargez WMF 4.0 : rendez-vous à :
   http://www.microsoft.com/en-us/download/details.aspx?id=40855
3. Choisir le fichier : Windows6.1-KB2819745-x86-MultiPkg.msu
4. Installez .NET Framework 4.5 et ensuite le ficher ci-dessus.

Console et environnement

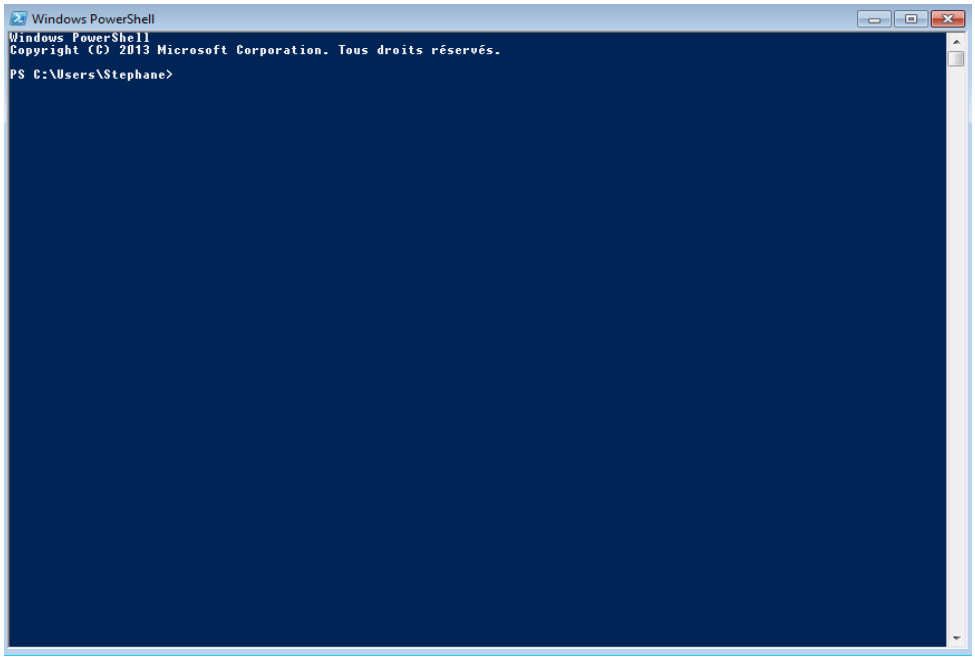
La console texte disponible avec powershell peut s'ouvrir de deux façons :

En cliquant windows powershell comme montré ci-dessous:



Windows Powershell ouvrira une fenêtre console alors que Windows Powershell ISE ouvrira l'environnement de développement graphique pour Powershell.

Voici à quoi ressemble la console texte :



Environnement intégré

Lorsque vous démarrez windows powershell ISE, vous obtenez un environnement de développement rudimentaire mais qui offre cependant toutes les fonctionnalités d'un éditeur de texte avec également accès à la console texte pour taper les applets de commandes et une autre portion de fenêtre qui montre les résultats. Voici ce à quoi ça ressemble :