

## Fiche 15 : Accès distant avec PowerShell

### 1. Introduction

La dernière version de Windows PowerShell v2 (RC) inclut une nouvelle version de la gestion à distance de Windows (WinRM 2.0 : Windows Remote Management). Ces deux éléments sont intégrés dans Windows 7 et Windows Server 2008 R2, et peuvent être installés dans Windows XP avec service pack 3 (voir <http://support.microsoft.com/kb/968929>).

Cette gestion distante permet d'exécuter des commandes sur un ou plusieurs ordinateurs distants à partir d'un seul ordinateur. PowerShell permet plusieurs moyens de connexion, interactif (1:1) ou un-à-plusieurs (1:n).

### 2. Démarrer le service WinRM

Sur les ordinateurs distants (serveurs), utiliser la commande *Enable-PSRemoting* pour démarrer le service WinRM.



```
Administrateur : Windows PowerShell
PS C:\> Enable-PSRemoting -Force
WinRM est déjà configuré pour recevoir des demandes sur cet ordinateur.
WinRM est déjà configuré pour la gestion à distance sur cet ordinateur.
```

**Rmq** : *-force* permet de ne pas répondre aux invites. Par défaut, vous êtes invité à confirmer chaque opération.

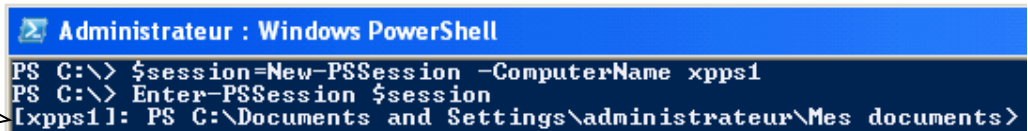
Dans Windows 7 et Windows server 2008 R2, pour utiliser cette commande, il faut lancer PowerShell en tant que administrateur (clic droit, exécuter en tant que ....)

Ce service utilise Kerberos par défaut pour l'authentification. Quand Kerberos n'est pas disponible (postes non intégrés à un domaine), WinRM utilise un transport HTTPS basé sur un certificat SSL installé sur le poste distant. Pour simplifier, cette fiche est donc réalisée avec un domaine Active Directory.

### 3. Connexion interactive (1:1)

Sur l'ordinateur client, exécuter l'instruction *\$session=New-PSSession -computername NomOrdinateur* et utiliser la commande *Enter-PSSession \$session* pour se connecter de manière interactive avec le poste distant.

Maintenant, les commandes seront exécutées dans la console distante de xpps1.



```
Administrateur : Windows PowerShell
PS C:\> $session=New-PSSession -ComputerName xpps1
PS C:\> Enter-PSSession $session
[xpps1]: PS C:\Documents and Settings\administrateur\Mes documents>
```

La commande *Exit-PSSession* permet de quitter la console distante et revenir à la console locale.

**Rmq** : Il est possible de spécifier un nom d'utilisateur pour les sessions avec le paramètre *-Credential*



```
Administrateur : Windows PowerShell
PS C:\> $session=New-PSSession -ComputerName xpps1 -Credential labo\administrateur
```

### 4. Accès à distance un-à-plusieurs (1:n)

L'accès à distance de PowerShell permet aussi d'exécuter une ou un ensemble de commandes simultanément sur plusieurs ordinateurs. La technique est très simple, il suffit de spécifier la liste des ordinateurs concernés avec le paramètre *-computername NomOrd1,NomOrd2,NomOrd3*.

Exemple avec un fichier texte : *\$session=New-PSSession -computername (Get-Content c:\listeOrd.txt)*

Pour exécuter une commande sur l'ensemble des postes, il faut utiliser : *Invoke-Command -Scriptblock {cmd}*

Exemple pour exécuter *IPconfig* sur chaque poste : *Invoke-Command -Scriptblock {ipconfig} -session \$session*



```
Administrateur : Windows PowerShell
PS C:\> $sessions=New-PSSession -ComputerName xp,xpps1
PS C:\> Invoke-Command -ScriptBlock {ipconfig} -Session $sessions
```

Cette commande affiche le résultat des deux *ipconfig* lancés à partir de chaque poste distant.

Il est possible d'utiliser *Invoke-Command* sans sessions ouvertes préalablement et de spécifier directement la liste des postes avec *-computername*, dans ce cas les sessions sont ouvertes au début et fermées juste après l'exécution.

Exemple : *Invoke-Command -Scriptblock {ipconfig} -computername NomOrdi1,NomOrdi2,NomOrdi3*

```
Administrateur : Windows PowerShell
PS C:\> Invoke-Command -ScriptBlock {ipconfig} -ComputerName xp,xpps1
```

En général, on crée une session uniquement lorsqu'on exécute une série de commandes sur l'ordinateur distant.

Par défaut, le nombre de connexions simultanées est de 32, cette valeur peut être modifiée avec le paramètre *-throttlimit* de la commande *Invoke-Command*.

Autres exemples avec *Invoke-command* :

Exécute le script local sur le poste distant : *Invoke-Command -filepath c:\scripts\test.ps1 -computerName NomOrdi1*

## 5. Tâches en arrière plan

L'exécution des commandes peut prendre du temps, même si elles s'exécutent de manière simultanée sur tous les postes. Dans ce cas, il est possible de laisser l'interface de commandes fonctionner en arrière plan avec le paramètre *-AsJob*.

Exemple 1 : *Invoke-Command -Scriptblock {ipconfig} -computername NomOrdi1,NomOrdi2,NomOrdi3 -AsJob*

```
Administrateur : Windows PowerShell
PS C:\> Invoke-Command -ScriptBlock {ipconfig} -ComputerName xp,xpps1 -AsJob

Id      Name      State      HasMoreData  Location      Command
--      -
1       Job1      En cours   True          xp,xpps1      ipconfig
```

Maintenant, les résultats sont stockés dans le cadre d'une tâche.

Pour voir les tâches en cours et leur état, on utilise : *Get-Job* avec éventuellement les paramètres de filtrage suivants :

- *id* : Numéro de la tâche,
- *Name* : Nom de la tâche
- *State* : état de la tâche (*Completed*, *Failed*, *Running*)

Exemple 1 suite : *Get-Job* suite à l'instruction précédente, l'état de la tâche est passé de *Running* à *Completed* :

```
PS C:\> Get-Job

Id      Name      State      HasMoreData  Location      Command
--      -
1       Job1      Terminée True          xp,xpps1      ipconfig
```

Exemple 2 : Ajouter une passerelle par défaut sur les postes distants (réalisée avec ISE, éditeur intégré à PowerShell)

```
test1passerelle.ps1 X
1 Invoke-Command -ScriptBlock {route add 0.0.0.0 mask 0.0.0.0 10.0.0.26} -ComputerName xp,xpps1 -AsJob
```

Cette méthode associe une seule tâche principale pour l'instruction *Invoke-Command*. Si un seul poste n'est pas joignable, la tâche principale est déclarée *Failed*. Exemple avec le poste *xp1* qui n'existe pas :

```
Administrateur : Windows PowerShell
PS C:\> Get-Job

Id      Name      State      HasMoreData  Location      Command
--      -
1       Job1      Failed     True          xp1,xpps1      ipconfig
```

Mais à cette tâche principale (dite aussi parent), sont associées autant de tâches enfants qu'il y a de poste.

Exemple pour voir les tâches enfants :

```
Administrateur : Windows PowerShell
PS C:\> Get-Job | select -Property ChildJobs,Location

ChildJobs                                     Location
-----
{Job2, Job3}                                xp1,xpps1
```

On peut afficher directement l'état d'une tâche enfant en spécifiant son nom (*Failed*, car xp1 non joignable) :

```
Administrateur : Windows PowerShell
PS C:\> Get-Job -Name job2

Id      Name      State      HasMoreData  Location      Command
--      -
2       Job2      Failed     False        xp1           ipconfig
```

Exemple : Afficher le(s) poste(s) (Location) dont l'état d'une tâche enfant est *Failed*. Ici, le poste xp1 n'existe pas :

```
Administrateur : Windows PowerShell
PS C:\> Get-Job -state failed | foreach <$_.childJobs> | Where-Object -FilterScript <$_.state -eq "Failed"> | select Location

Location
-----
xp1
```

Exemple 2 : Idem, mais affiche en plus la raison de l'échec :

```
Administrateur : Windows PowerShell
PS C:\> Get-Job -state failed | foreach <$_.childJobs> | foreach <if <$_.state -eq "Failed"><$_.Location;$_.JobStateInfo.reason >>
xp1
La connexion au serveur distant a échoué avec le message d'erreur suivant : WinRM ne peut pas traiter la demande. L'erreur suivante
```