JEA

JEA (Just Enough Administration) est une fonctionnalité de Windows PowerShell qui permet de limiter les privilèges des administrateurs en leur donnant juste les droits nécessaires pour effectuer des tâches spécifiques, sans leur accorder un accès complet..

Créer le fichier de configuration de session

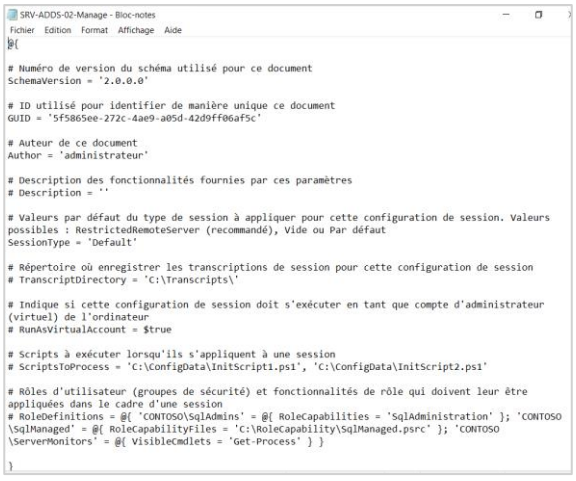
On se connecte sur notre serveur AD, on ouvre PowerShell et on entre la commande : New-PSSessionConfigurationFile -Path 'C:\Program Files\WindowsPowerShell\SRV-ADDS-01-Manage.pssc'

Cela nous crée un fichier nommé SRV-ADDS-01-Manage.pssc où PSSC correspond à **P**ower**S**hell **S**ession **C**onfiguration.

Ce sera le fichier de configuration de notre session.

On peut **l'ouvrir** directement dans le dossier cité dans la commande ou utiliser la commande : notepad.exe 'C:\Program Files\WindowsPowerShell\SRV-ADDS-01-Manage.pssc'

On obtient le fichier suivant :



Les lignes surlignées sont celles que j'ai modifiées.

- J'ai remplacé la valeur "**Default**" par "**RestrictedRemoteServer**". Recommandé par Microsoft, cela va restreindre directement les commandes accessibles.
  Avec ce mode, l'utilisateur peut utiliser seulement les commandes suivantes :
  - Clear-Host
  - Exit-PSSession
  - Get-Command
  - Get-FormatData
  - Get-Help

- Measure-Object
- Out-Default
- Select-Object

- **TranscriptDirectory :** chaque session distante aura son propre journal de transcription. Ces fichiers de transcription doivent être stockés dans un dossier : précisez le chemin de ce dossier et pensez à le créer.
- **RunAsVirtualAccount :** lorsque cette option est à $true (vrai), l'utilisateur bénéficie d'un compte virtuel qui est administrateur local du serveur. Autrement dit, les actions que l'utilisateur va exécuter seront effectuées avec des droits "admin" sur le serveur même si lui n'est pas administrateur du serveur.
- **RoleDefinitions :** je précise le groupe de sécurité Active Directory qui est autorisé à utiliser cette session, puis le nom du rôle associé.

On sauvegarde le fichier et on lance la commande : Test-PSSessionConfigurationFile 'C:\Program Files\WindowsPowerShell\SRV-ADDS-01-Manage.pssc'

Si elle nous renvoie le résultat True, c'est qu'il n'y a pas d'erreur de syntaxe dans notre fichier.

Créer le fichier de rôle psrc

On commence par créer un dossier "RoleCapabilities" sur notre serveur pour stocker l'ensemble des fichiers de configuration de nos rôles.

PowerShell sur l'AD : New-Item -Path 'C:\Program Files\WindowsPowerShell\Modules\JEA\RoleCapabilities' -ItemType Directory

Le dossier étant créé, on va créer notre fichier "Support-IT_Manage.psrc" tout en sachant que l'extension PSRC correspond à PowerShell Session Role Capabilities. Ce fichier doit être nommé de la même façon que le rôle, c'est impératif.

New-PSRoleCapabilityFile -Path 'C:\Program Files\WindowsPowerShell\Modules\JEA\RoleCapabilities\Support-IT_Manage.psrc'

On ouvre le fichier. notepad.exe 'C:\Program Files\WindowsPowerShell\Modules\JEA\RoleCapabilities\Support-IT_Manage.psrc'

On obtient ce fichier :

Support-IT_Manage.psrc - Bloc-notes

Fichier Edition Format Affichage Aide

```
@{

# ID utilisé pour identifier de manière unique ce document
GUID = '149a740d-3b13-4f39-8362-5a782199da80'

# Auteur de ce document
Author = 'administrateur'

# Description des fonctionnalités fournies par ces paramètres
# Description = ''

# Company associated with this document
CompanyName = 'Inconnu'

# Instruction de copyright pour ce document
Copyright = '(c) 2025 administrateur. Tous droits réservés.'

# Modules à importer lorsqu'ils s'appliquent à une session
# ModulesToImport = 'MyCustomModule', @{ ModuleName = 'MyCustomModule'; ModuleVersion = '1.0.0.0'; GUID = '4d30d5f0-cb16-4898-812d-f20a6c596bdf' }

# Alias à rendre visibles lorsqu'ils s'appliquent à une session
# VisibleAliases = 'Item1', 'Item2'

# Cmdlets à rendre visibles lorsqu'elles s'appliquent à une session
# VisibleCmdlets = 'Invoke-Cmdlet1', @{ Name = 'Invoke-Cmdlet2'; Parameters = @{ Name = 'Parameter1'; ValidateSet = 'Item1', 'Item2' }, @{ Name = 'Parameter2'; ValidatePattern = 'L*' } }

# Fonctions à rendre visibles lorsqu'elles s'appliquent à une session
# VisibleFunctions = 'Invoke-Function1', @{ Name = 'Invoke-Function2'; Parameters = @{ Name = 'Parameter1'; ValidateSet = 'Item1', 'Item2' }, @{ Name = 'Parameter2'; ValidatePattern = 'L*' } }

# Commandes externes (scripts et applications) à rendre visible lorsqu'elles s'appliquent à une session
# VisibleExternalCommands = 'Item1', 'Item2'

# Fournisseurs à rendre visibles lorsqu'ils s'appliquent à une session
# VisibleProviders = 'Item1', 'Item2'

# Scripts à exécuter lorsqu'ils s'appliquent à une session
# ScriptsToProcess = 'C:\ConfigData\InitScript1.ps1', 'C:\ConfigData\InitScript2.ps1'

# Alias à définir lorsqu'ils s'appliquent à une session
# AliasDefinitions = @{ Name = 'Alias1'; Value = 'Invoke-Alias1'}, @{ Name = 'Alias2'; Value = 'Invoke-Alias2'}

# Fonctions à définir lorsqu'elles s'appliquent à une session
# FunctionDefinitions = @{ Name = 'MyFunction'; ScriptBlock = { param($MyInput) $MyInput } }

# Variables à définir lorsqu'elles s'appliquent à une session
# VariableDefinitions = @{ Name = 'Variable1'; Value = { 'Dynamic' + 'InitialValue' } }, @{ Name = 'Variable2'; Value = 'StaticInitialValue' }

# Variables d'environnement à définir lorsqu'elles s'appliquent à une session
# EnvironmentVariables = @{ Variable1 = 'Value1'; Variable2 = 'Value2' }

# Fichiers de type (.ps1xml) à charger lorsqu'ils s'appliquent à une session
# TypesToProcess = 'C:\ConfigData\MyTypes.ps1xml', 'C:\ConfigData\OtherTypes.ps1xml'

# Fichiers de format (.ps1xml) à charger lorsqu'ils s'appliquent à une session
# FormatsToProcess = 'C:\ConfigData\MyFormats.ps1xml', 'C:\ConfigData\OtherFormats.ps1xml'

# Assemblys à charger lorsqu'ils s'appliquent à une session
# AssembliesToLoad = 'System.Web', 'System.OtherAssembly, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a'

}
```

Voici la version modifiée :

*Support-IT_Manage.psrc - Bloc-notes

Fichier Edition Format Affichage Aide

```
@{

# ID utilisé pour identifier de manière unique ce document
GUID = '149a740d-3b13-4f39-8362-5a782199da80'

# Auteur de ce document
Author = 'administrateur'

# Description des fonctionnalités fournies par ces paramètres
# Description = ''

# Company associated with this document
CompanyName = 'Inconnu'

# Instruction de copyright pour ce document
Copyright = '(c) 2025 administrateur. Tous droits réservés.'

# Modules à importer lorsqu'ils s'appliquent à une session
# ModulesToImport = 'MyCustomModule', @{ ModuleName = 'MyCustomModule'; ModuleVersion = '1.0.0.0'; GUID = '4d30d5f0-cb16-4898-812d-f20a6c596bdf' }

# Alias à rendre visibles lorsqu'ils s'appliquent à une session
# VisibleAliases = 'Item1', 'Item2'

# Cmdlets à rendre visibles lorsqu'elles s'appliquent à une session
# VisibleCmdlets = 'Invoke-Cmdlet1', @{ Name = 'Invoke-Cmdlet2'; Parameters = @{ Name = 'Parameter1'; ValidateSet = 'Item1', 'Item2' }, @{ Name = 'Parameter2'; ValidatePattern = 'L*' } }
VisibleCmdlets = 'Restart-Computer', 'Get-Service', @{ Name = 'Restart-Service'; Parameters = @{ Name = 'Name'; ValidateSet = 'DHCPServer' }}

# Fonctions à rendre visibles lorsqu'elles s'appliquent à une session
# VisibleFunctions = 'Invoke-Function1', @{ Name = 'Invoke-Function2'; Parameters = @{ Name = 'Parameter1'; ValidateSet = 'Item1', 'Item2' }, @{ Name = 'Parameter2'; ValidatePattern = 'L*' } }

# Commandes externes (scripts et applications) à rendre visible lorsqu'elles s'appliquent à une session
# VisibleExternalCommands = 'Item1', 'Item2'
VisibleExternalCommands = 'C:\Windows\System32\ping.exe'

# Fournisseurs à rendre visibles lorsqu'ils s'appliquent à une session
# VisibleProviders = 'Item1', 'Item2'

# Scripts à exécuter lorsqu'ils s'appliquent à une session
# ScriptsToProcess = 'C:\ConfigData\InitScript1.ps1', 'C:\ConfigData\InitScript2.ps1'

# Alias à définir lorsqu'ils s'appliquent à une session
# AliasDefinitions = @{ Name = 'Alias1'; Value = 'Invoke-Alias1'}, @{ Name = 'Alias2'; Value = 'Invoke-Alias2'}

# Fonctions à définir lorsqu'elles s'appliquent à une session
# FunctionDefinitions = @{ Name = 'MyFunction'; ScriptBlock = { param($MyInput) $MyInput } }

# Variables à définir lorsqu'elles s'appliquent à une session
# VariableDefinitions = @{ Name = 'Variable1'; Value = { 'Dynamic' + 'InitialValue' } }, @{ Name = 'Variable2'; Value = 'StaticInitialValue' }

# Variables d'environnement à définir lorsqu'elles s'appliquent à une session
# EnvironmentVariables = @{ Variable1 = 'Value1'; Variable2 = 'Value2' }

# Fichiers de type (.ps1xml) à charger lorsqu'ils s'appliquent à une session
# TypesToProcess = 'C:\ConfigData\MyTypes.ps1xml', 'C:\ConfigData\OtherTypes.ps1xml'

# Fichiers de format (.ps1xml) à charger lorsqu'ils s'appliquent à une session
# FormatsToProcess = 'C:\ConfigData\MyFormats.ps1xml', 'C:\ConfigData\OtherFormats.ps1xml'

# Assemblys à charger lorsqu'ils s'appliquent à une session
# AssembliesToLoad = 'System.Web', 'System.OtherAssembly, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a'

}
```

Pour rappel, nous souhaitons que les membres du groupe "Support-IT" soient autorisés à :

Redémarrer le service DHCP

Restart-Service -Name DHCPServer

Redémarrer le serveur

Restart-Computer

Lister les services actifs sur le serveur

Get-Service

Réaliser un ping à partir de ce serveur.

Ping

Il faut donc que l'on autorise les cmdlets "Restart-Computer" et "Get-Service". Ensuite, il faut que l'on permette à l'utilisateur de redémarrer le service "DHCPServer ».

Donc, la commande PowerShell à autoriser est :

Restart-Service -Name DHCPServer

Au sein de la configuration, voici comment nous allons transposer cette commande :

@{ Name = 'Restart-Service'; Parameters = @{ Name = 'Name'; ValidateSet = 'DHCPServer' }}

On précise que l'on accepte seulement "DHCPServer" comme valeur pour le paramètre "-Name". Si l'on veut autoriser plusieurs valeurs, voici la syntaxe (pour autoriser aussi la valeur "NTDS") :

@{ Name = 'Restart-Service'; Parameters = @{ Name = 'Name'; ValidateSet = 'DHCPServer', 'NTDS'}}

Finalement, voici la ligne complète pour autoriser les trois cmdlets :

VisibleCmdlets = 'Restart-Computer', 'Get-Service', @{ Name = 'Restart-Service'; Parameters = @{ Name = 'Name'; ValidateSet = 'DHCPServer' }}

Pour autoriser le "ping" (car oui il est bloqué lui aussi même si c'est une commande basique !), il faut configurer l'option "VisibleExternalCommands". On précise tout simplement le chemin complet vers "ping.exe", ce qui donne :

VisibleExternalCommands = 'C:\Windows\System32\ping.exe'

Nous **créons notre groupe Support-IT et son OU :**

New-ADOrganizationalUnit -Name "GrpSupport" -Path "DC=roncenoir,DC=local"

New-ADGroup -Name "Support-IT" -SamAccountName "Support-IT" -GroupScope Global -GroupCategory Security -Path "OU=GrpSupport,DC=roncenoir,DC=local"

Nous devons **enregistrer cette session PowerShell** pour qu'elle soit utilisable. Le cmdlet Register-PSSessionConfiguration doit être utilisé. On va **spécifier le nom de la configuration "Support-IT"** et le **chemin vers le fichier PSSC : SRV-ADDS-01-Manage.pssc.**

Register-PSSessionConfiguration –NameSupport-IT -Path 'C:\Program Files\WindowsPowerShell\SRV-ADDS-01-Manage.pssc'

```
PS C:\Users\administrateur.RONCENOIR> Register-PSSessionConfiguration -Name Support-IT -Path 'C:\Program Files\WindowsPowerShell\SRV-ADDS-02-Manage.pssc'
AVERTISSEMENT : Register-PSSessionConfiguration peut avoir besoin de redémarrer le service WinRM si l'inscription d'une configuration utilisant ce nom a récemment été annulée, et que
certaines structures de données système peuvent encore se trouver en cache. Dans ce cas, un redémarrage de WinRM peut être nécessaire.
Toutes les sessions WinRM connectées à des configurations de session Windows PowerShell, telles que Microsoft.PowerShell et les configurations de session créées avec l'applet de commande
Register-PSSessionConfiguration, sont déconnectées.


   WSManConfig : Microsoft.WSMan.Management\WSMan::localhost\Plugin

Type            Keys                                Name
----            ----                                ----
Container       {Name=Support-IT}                   Support-IT
AVERTISSEMENT : Set-PSSessionConfiguration peut avoir besoin de redémarrer le service WinRM si l'inscription d'une configuration utilisant ce nom a récemment été annulée, et que certaines
structures de données système peuvent encore se trouver en cache. Dans ce cas, un redémarrage de WinRM peut être nécessaire.
Toutes les sessions WinRM connectées à des configurations de session Windows PowerShell, telles que Microsoft.PowerShell et les configurations de session créées avec l'applet de commande
Register-PSSessionConfiguration, sont déconnectées.
AVERTISSEMENT : Register-PSSessionConfiguration peut avoir besoin de redémarrer le service WinRM si l'inscription d'une configuration utilisant ce nom a récemment été annulée, et que
certaines structures de données système peuvent encore se trouver en cache. Dans ce cas, un redémarrage de WinRM peut être nécessaire.
Toutes les sessions WinRM connectées à des configurations de session Windows PowerShell, telles que Microsoft.PowerShell et les configurations de session créées avec l'applet de commande
Register-PSSessionConfiguration, sont déconnectées.
```

On redémarre ensuite le service WinRM : Restart-Service WinRM

On vérifie que notre rôle apparaît dans les permissions avec : Get-PSSessionConfiguration | Format-Table Name, Permission

```
PS C:\Users\administrateur.RONCENOIR> Get-PSSessionConfiguration | Format-Table Name, Permission

Name                                Permission
----                                ----------
microsoft.powershell                AUTORITE NT\INTERACTIF AccessAllowed, BUILTIN\Administrateurs AccessAllowed, BUILTIN\Utilisateurs de gestion à distance AccessAllowed
microsoft.powershell.workflow       BUILTIN\Administrateurs AccessAllowed, BUILTIN\Utilisateurs de gestion à distance AccessAllowed
microsoft.powershell32              AUTORITE NT\INTERACTIF AccessAllowed, BUILTIN\Administrateurs AccessAllowed, BUILTIN\Utilisateurs de gestion à distance AccessAllowed
microsoft.windows.servermanagerworkflows AUTORITE NT\INTERACTIF AccessAllowed, BUILTIN\Administrateurs AccessAllowed
Support-IT                          RONCENOIR\support-it AccessAllowed
```

On teste l'ouverture d'une session

Pour tester, on se connecte sur un pc appartenant au domaine. Ici, j'utilise un client Windows10 et je m'y connecte avec un compte AD appartenant au domaine et membre du groupe Support-IT. **Dans cet exemple, j'ai donné ce rôle à Maven RONCENOIR et Idaryn DUNMER, Directrice et bras droit de l'entreprise. Leurs identifiants AD sont m.roncenoir et i.dunmer .**

Dans PowerShell, j'ouvre une session avec la commande : Enter-PSSession -ComputerName SRV-ADDS-01 -ConfigurationName Support-IT

```
PS C:\Users\i.dunmer> Enter-PSSession -ComputerName ADSecure2 -ConfigurationName Support-IT
[ADSecure2]: PS>
```

Je réalise la même installation et manipulation sur le second ADSecure afin que les membres aient la main sur les deux contrôleurs de domaine.