PowerShell Extension

07.04.2020 – v1.1.0

Überblick

Zur Extension

Weil die Powershell und .NET viel Potenzial haben, aber die administrativen Aufgaben nicht out oft he box abgedeckt werden können, ist ipx entstanden. Der primäre Anreiz war eine bessere Verwaltung und Bearbeitung von Dateibeständen, mittlerweile finden sich aber auch andere vermisste Funktionen im Modul.

Ipx ist skriptbasiert und hat somit einen offenen Code. Die Entwicklung erfolgt in einem Firmen-Domänennetzwerk mit Praxiseinsatz auf Servern und Clients. Inspiration, Hintergrundwissen und Codebeispiele geben:

- Powershell Community Extension PSCX
- Dr.Tobias Weltner: Scripting mit Windows PowerShell 3.0 - Der Workshop
- Holger Schwichtenberg: Windows Powershell 4.0 - Das Praxisbuch
- Heise Zeitschriften Verlag, diverse c´t wie 3/2015 ISE-Erweiterungen

Durch die skriptbasierte Struktur kann ipx zentral im Firmennetzwerk mit wenig Aufwand für alle Rechner bereitgestellt werden.

Haftungsausschluss

Zwar teste ich die Cmdlets soweit es mir möglich ist, ich kann jedoch nicht jeden Fehler ausschließen. Auch kann ich keinen Einfluss auf lokale Gegebenheiten nehmen, die Auswirkungen auf die Skriptausführung haben, beispielsweise Virenscanner. Auch bin ich machtlos gegen falsche Benutzung durch die es zum versehentlichen Datenverlust kommt.

Es obliegt dem Anwender die gebotenen Funktionen auf Herz und Nieren zu testen und sie erst dann im Praxiseinsatz zu nutzen. Ich hafte nicht für durch den Benutzer verursachte Fehler oder nicht absehbare Systemumgebungen, die sich vollständig anders verhalten als mit üblichen Windows-Installationen nachvollziehbar zu testen ist.

Lizenz

Das Modul wird mit einer Creative Commons 4.0 Namensnennung share alike Lizenz (CC BY SA) angeboten, kann also in kommerziellen Umgebungen und in eigenen Projekten genutzt werden.

Anforderungen

- Windows 7, 8, 8.1, 10, Server 2008R2, 2012, 2012R2 mit
- Powershell 3.0, 4.0 und 5.0 bzw. das zugehörige Management Framework
- .NET 4.0 oder neuer, sicherlich auch .net Core ab 3.0 mit installierten oder mittlerweile verfügbaren Erweiterungen für z.B. Out-Gridview.

- Für einige Funktionen die PowerShell Community Extension 3.2.0 (oder neuer?)
- Für die zentrale Verwaltung eine Netzwerkfreigabe sowie ActiveDirectory mit Gruppenrichtlinien

Struktur

Für den reinen Betrieb sind nur die beiden Ordner Modules mit Inhalt und Scripts notwendig. Ini und logs empfehlen sich für den Alltag, um beispielsweise Auswertungen zentral abzulegen.

```
\WindowsPowerShell
|--ini
|--Logs
|--Modules
|  |--ipx
|  |  |--docs
|  |  |--exe
|  |  |--profiles
|  |  |--xaml
|  |  |--ModuleLoader.psm1
|  |  |--ipx.psd1
|  |  \--ipx.ps1
\--Scripts
```

Ordner

Ordner	Funktion
docs	Dokumentation des Moduls
exe	Enthält notwendige Programme für das Modul
profiles	Beinhaltet die Dummy Profile für Powershell und ISE für die Einbindung am Client.
xaml	Enthält die WPF xaml-Dateien für die Cmdlets mit grafischer Oberfläche.

Dateien

Datei	Funktion
ipx.psd1	Moduldatei mit Basisinformationen für die Definition des Moduls selbst.
ModuleLoader.psm1	Läd alle Befehle in den *.ps1 Dateien im Modulordner für den Gebrauch.
ipx.ps1	Alle Cmdlets und Funktionen des Moduls.
exe\7za.exe	7zip Kommandozeilenversion.

Installation

Die Installation unterscheidet sich minimal zwischen Domänennetzwerken und reinen Arbeitsgruppen.

Domänennetzwerk.

1] Auf einem Dateiserver oder NAS werden die notwendigen Ordner ähnlich der Struktur angelegt. Im ipx Ordner finden sich nur die eigenen Ordner, die Ebenen darüber müssen selbst in den eigenen Bestand eingepflegt werden. Im Installationsbeispiel heißt der Server fileserver01 und die Freigabe Daten\Repository\WindowsPowershell.

2] Die profile_dummy.ps1 wird aus dem Ordner profiles nach Scripts kopiert und in profile.ps1 umbenannt. Jetzt die Datei öffnen und die notwendige Anpassungen durchführen: In Zeile 6 wird der Serverpfad eingetragen, in diesem Beispiel wäre das also \\fileserver01\Daten\Repository\WindowsPowerShell.

```
1  <#
2      .Synopsis
3          Erweitert die lokale Powershell Session um die verfügbaren Module
4  #>
5
6   [string]$MyModuls = "\\fileserver01\Daten\Repository\WindowsPowerShell"
7
8  if ( Test-Path -path $MyModuls\Modules) {
9      $env:PSModulePath += ";$MyModuls\Modules"
10     Get-ChildItem -Path $MyModuls\Modules -Directory | ForEach-Object {
11     Import-Module $_
12
13     if (Get-Module -ListAvailable -Name $_.Name) {
14             Write-Host $_.Name -ForegroundColor Green
15         } else {
16             Write-Host $_.Name.ToString() "konnte nicht geladen werden" -ForegroundColor Red
17         }
18     }
19     $env:Path += ";$MyModuls\Skripts"
20 }
21 else {
22     $env:PSModulePath += ";$env:userprofile\Documents\MyPowershell\Modules"
23     Get-ChildItem -Path $env:userprofile\Documents\MyPowershell\Modules -Directory | ForEach-Object {
24     Import-Module $_
25
26     if (Get-Module -ListAvailable -Name $_.Name) {
27             Write-Host $_.Name -ForegroundColor Green
28         } else {
29             Write-Host $_.Name.ToString() "konnte nicht geladen werden" -ForegroundColor Red
30         }
31     }
32     $env:Path += ";$env:userprofile\Documents\MyPowershell\Skripts"
33 }
```

3] Für die Verteilung der profile.ps1 wird eine Gruppenrichtlinie (GPO) für die Computerkonfiguration angelegt:
`Einstellungen > Windows Einstellungen > Dateien > Neu`
`Quelldatei(en)`
`\\fileserver01\Daten\Repository\WindowsPowerShell\Scripts\profile.ps1`
`Zieldatei %windir%\System32\WindowsPowerShell\v1.0\profile.ps1`
`Gemeinsam: Verarbeitungstyp ist nicht Hintergrund` (Kopiert nur bei Login)
Die Richtlinie wird auf alle OUs angewendet die relevante PCs enthalten.

4] Für die Sicherheitsrichtlinie der Powershell wird eine zweite GPO angelegt:
`Benutzerkonfiguration > Administrative Vorlagen > Windows > Windows Komponenten > Windows Powershell:`

`Skriptausführung aktivieren: Lokale und Remote signierte Skripts`
Die GPO wird auf alle Benutzer angewendet die mit der Powershell arbeiten.

5] Speziell bei Servern muss die verstärkte Sicherheitsrichtlinie für den IE deaktiviert werden:
`Server-Manager öffnen > Lokaler Server > Verstärkte Sicherheitskonfiguration für IE`
`Administratoren: Aus`
`Benutzer: Ein.`
Ohne diese Deaktivierung kommt es trotz der GPOs zu Fehlern beim Modulimport.

Arbeitsgruppe ohne Domäne

1] Auf einem Dateiserver oder NAS werden die notwendigen Ordner ähnlich der Struktur angelegt. Im ipx Ordner finden sich nur die eigenen Ordner, die Ebenen darüber müssen selbst in den eigenen Bestand eingepflegt werden. Im Installationsbeispiel heißt der Server fileserver01 und die Freigabe Daten\Repository\WindowsPowershell.

2] Die profile_dummy.ps1 wird aus dem Ordner profiles nach Scripts kopiert und in profile.ps1 umbenannt. Jetzt die Datei öffnen und die notwendige Anpassungen durchführen: In Zeile 6 wird der Serverpfad eingetragen, in diesem Beispiel wäre das also \\fileserver01\Daten\Repository\WindowsPowerShell.

```
1  <#
2      .Synopsis
3          Erweitert die lokale Powershell Session um die verfügbaren Module
4  #>
5
6   [string]$MyModuls = "\\fileserver01\Daten\Repository\WindowsPowerShell"
7
8  if ( Test-Path -path $MyModuls\Modules) {
9      $env:PSModulePath += ";$MyModuls\Modules"
10     Get-ChildItem -Path $MyModuls\Modules -Directory | ForEach-Object {
11     Import-Module $_
12
13     if (Get-Module -ListAvailable -Name $_.Name) {
14             Write-Host $_.Name -ForegroundColor Green
15         } else {
16             Write-Host $_.Name.ToString() "konnte nicht geladen werden" -ForegroundColor Red
17         }
18     }
19     $env:Path += ";$MyModuls\Skripts"
20 }
21 else {
22     $env:PSModulePath += ";$env:userprofile\Documents\MyPowershell\Modules"
23     Get-ChildItem -Path $env:userprofile\Documents\MyPowershell\Modules -Directory | ForEach-Object {
24     Import-Module $_
25
26     if (Get-Module -ListAvailable -Name $_.Name) {
27             Write-Host $_.Name -ForegroundColor Green
28         } else {
29             Write-Host $_.Name.ToString() "konnte nicht geladen werden" -ForegroundColor Red
30         }
31     }
32     $env:Path += ";$env:userprofile\Documents\MyPowershell\Skripts"
33 }
```

3] Die Powershell wird explizit mit Administrationsrechten gestartet. Die Sicherheit wird mit `Set-ExecutionPolicy RemoteSigned` gesetzt. Anschließend die Powershell ohne Adminrechte neu starten und Einstellung mit `Get-ExecutionPolicy` überprüfen. Die Powershell bleibt noch geöffnet.

Bei Rechnern mit deaktivierte Benutzerkontensteuerung sollte die Powershell grundsätzlich mit Adminrechten laufen. Kann nicht empfohlen werden.

4] In der Powershell den Befehl `explorer $pshome` ausführen, es sollte sich das Installationsverzeichnis der Powershell öffnen. Die angepasste profile.ps1 wird dort reinkopiert, das muss von einem lokalen Laufwerk geschehen. Bei Netzlaufwerken sperrt sich gerne Windows gegen den Zugriff.

5] Jetzt die Powershell neu starten und checken ob alles geklappt hat.

Ganz ohne Netzwerk

Der Ablauf ähnelt dem in einem Netzwerk ohne Domäne. In der profile.ps1 kann der komplette `if`-Zweig entfernt werden (Zeile 6-21 und 33), es muss nur das zwischen `else { }` übrig bleiben. Und sofern die Ordner wie dort verzeichnet angelegt wurden (Zeile 22), müssen auch die Pfade nicht angepasst werden.

```
1  <#
2      .Synopsis
3          Erweitert die lokale Powershell Session um die verfügbaren Module
4  #>
5
6    [string]$MyModuls = "\\fileserver01\Daten\Repository\WindowsPowerShell"
7
8  if ( Test-Path -path $MyModuls\Modules) {
9      $env:PSModulePath += ";$MyModuls\Modules"
10     Get-ChildItem -Path $MyModuls\Modules -Directory | ForEach-Object {
11     Import-Module $_
12
13     if (Get-Module -ListAvailable -Name $_.Name) {
14             Write-Host $_.Name -ForegroundColor Green
15         } else {
16             Write-Host $_.Name.ToString() "konnte nicht geladen werden" -ForegroundColor Red
17         }
18     }
19     $env:Path += ";$MyModuls\Skripts"
20 }
21 else {
22     $env:PSModulePath += ";$env:userprofile\Documents\MyPowershell\Modules"
23     Get-ChildItem -Path $env:userprofile\Documents\MyPowershell\Modules -Directory | ForEach-Object {
24     Import-Module $_
25
26     if (Get-Module -ListAvailable -Name $_.Name) {
27             Write-Host $_.Name -ForegroundColor Green
28         } else {
29             Write-Host $_.Name.ToString() "konnte nicht geladen werden" -ForegroundColor Red
30         }
31     }
32     $env:Path += ";$env:userprofile\Documents\MyPowershell\Skripts"
33 }
```

Deinstallation

Gibt es nicht wie bei bekannten Programmen anderer Hersteller. Es werden keine Einträge in der Registry erzeugt, sondern das Modul bei jedem Start der Powershell oder ISE neu eingelesen. Sobald also das Modul gelöscht, die profile.ps1 entfernt und etwaig die ExecutionPolicy zurückgesetzt wurde, ist das Modul nicht mehr verfügbar.

Cmdlets

Die Namen der Cmdlets versuchen sich an die Best Practices von Microsoft zu halten. Wo es sinnvoll ist, wurden die Befehle auch fit für die Pipeline gemacht. Ich verwende nie positionale Parameter, alle sind über ihren Namen anzusprechen. Die aktuelle Liste der Funktionen kann über `Get-ipxCommandlist -Name ipx` abgerufen werden. Auch versuche ich die Funktionen vollständig zu dokumentieren, `Get-Help` sollte also immer recht umfangreiche Ergebnisse liefern. Um einen Konflikt mit anderen Modulen zu vermeiden, ist allen Befehlsnamen das ipx vorangestellt.

FAQ

? Wieso heißt der lokale Ordner MyPowershell und nicht auch WindowsPowerShell?

! WindowsPowerShell ist der Standardpfad für benutzerspezifische Module und Skripte. Für Testzwecke und um es auseinanderzuhalten ist der schlichtweg anders benannt. `New-ipxCustomModule` schreibt ebenfalls in den Standardpfad, das gäbe Chaos.

? Warum gibt es keinen schönen Installer der einem die Arbeit beim Installieren abnimmt?

! Mehrere Antworten:

1. Ich kann noch kein C# in diesem Umfang. :D
2. Ich müsste alles digital signieren damit es ohne Anpassungen läuft.
3. Die Verteilung und Anpassung geht skriptbasiert in einer Domäne einfacher. Updates sind auch sofort aktiv ohne Updateprozedur auf jedem Rechner

? Im Domänennetzwerk wird die profile.ps1 nicht auf die Rechner verteilt.

! Wenn gpresult ausgibt, dass die Richtlinie angewendet wurde, liegt es mit hoher Wahrscheinlichkeit an den Zugriffsrechten. Seit einem Windows Update im Herbst 2016 wird die GPO in der Sicherheitsumgebung des ausführenden Rechners durchgeführt. Für die Netzwerkfreigabe muss deshalb zwingend ein `Leserecht` für die Gruppe `Authentifizierte Benutzer` erteilt werden. Standardmäßig besitzt die SYSVOL-Freigabe des AD dieses Recht, die profile.ps1 könnte also auch von dort ausgerollt werden. Das muss im eigenen Sicherheitskonzept entschieden werden.

? Die Powershell ISE lässt mich beim Entwickeln kein Script im Modul ausführen, obwohl es in der Shell selbst funktioniert und läuft.

! Das hängt mit der ExecutionPolicy zusammen und wie das Modul eingebunden wird. Als Benutzer bekommt man Netzwerkfreigaben für gewöhnlich als Laufwerk mit Buchstaben eingebunden. Die profile.ps1 fügt in Zeile 9 den UNC-Pfad, also \\fileserver\Daten..., und in Zeile 19 den Pfad zu den Skripten als Umgebungsvariable ein. Ein Netzlaufwerk wie S:\ entspricht keinem der beiden Pfade.

a) Man legt sich im Explorer einen Favoriten auf den UNC Pfad an, also \\fileserver... und öffnet die Skripte zum entwickelt darüber, oder
b) Da die Rechner seit Windows 8 ohnehin nicht mehr in jedem Fall der Domäne vertrauen, greift eine GPO die auch bei den reinen Freigaben nötig wird. Es wird eine neue GPO angelegt, auf alle notwendigen Rechner angewendet und folgendes eingestellt:
1. `Computerkonfiguration\Richtlinien\Administrative Vorlagen\Windows Komponenten\Internet Explorer\Internetsystemsteuerung\Sicherheitsseite: Liste der Site zu Zonenzuweisung: *.domain.xyz | 1`.
2. `Computerkonfiguration\Richtlinien\Administrative Vorlagen\Windows Komponenten\Internet ExplorerInternetsystemsteuerung\Sicherheitsseite\Intranetzone: Ziehen und Ablegen oder Kopieren und Einfügen von Dateien zulassen, Aktivieren`.
Damit vertrauen alle Rechner allen lokalen Domänen-Server oder -Clients. Ebenfalls verschwindet damit der Hinweis, wenn Dateien von z.B. einem DFS-Laufwerk auf den eigenen PC kopiert werden.

? Was fange ich mit dem Scripts-Ordner an?

! Die Cmdlets sind allgemein geschrieben. Im administrativen Skriptbetrieb existieren jedoch häufig fertige bat-Skripte mit einem festen Ablauf was gemacht werden soll. Der Scripts-Ordner stellt eine zentrale Stelle dar um solche Arbeitsskripte zu entwerfen und anzuwenden. Beispielsweise kann ein Backupskript mit `Copy-ipxBackupItem -Source … -Destination … -logpath …` entwickelt werden. Der ausführende Server ruft dann über die Windows Aufgabenplanung direkt `powershell.exe -noninteractive -command „skript.ps1“` auf und führt dies aus. Der ini und logs-Ordner haben einen ähnlichen Hintergrund, es können zentral alle Informationen gehalten werden.

? Wozu gibt es zwei Teile in der profile.ps1?

! Für alle Kollegen mit Laptop ohne DirectAccess/VPN oder schlichtweg keinem Internet. Die profile.ps1 versucht zuerst den Server anzusprechen und das Modul von dort zu laden. Wenn das nicht klappt, wird auf den lokalen Pfad zurückgegriffen. Wird auch dort nichts gefunden, ist das Modul schlicht nicht verfügbar. Vom Prinzip lässt sich das Modul so auch ohne Netzwerkzugriff benutzen.

? Die Ladezeit des Moduls wird immer länger.

! Skriptbasierte Module werden dateiweise importiert. Pro einzulesender Datei dauert das etwa 0,15 Sekunden. Bei zehn Dateien sind es dann schon 1,5 Sekunden. Zwar ist das entwickeln mit mehreren kleineren Dateien angenehmer und übersichtlicher, der Import beim Start der Powershell verzögert sich aber zunehmend.