PowerShell statt CMD

Die PowerShell-Exploration eines NICHT-Admins
in die Konsolen-Niederung,
zu den weiten Script-Feldern,
auf die Remote-Fernsicht und
zur .NET-Verwandtschaft.

Teil 1 PowerShell Konzepte

Themen

- Mein Einstieg

- Konsole
- Script
- Remote
- .NET-Verwandtschaft

- Ende

Motivation

Mein Berührungspunkt zu PowerShell

Anweisungen zum ***Update von BEFOmobil***, der Befo „Windows App“

1. Starten:	Windows PowerShell.exe
2. Wechseln auf:	V:
3. Wechseln ins Install.-Verz.:	cd \Referenz\App
4. Wechseln ins Betriebssys.-Verz:	cd ARM (für Win RT 8.1), cd x86 (sonst)
5. Zwei Befehle ausführen:	import-module appx
	add-appxpackage BEFOmobil.appx

Ankündigung: „Das wird zukünftig mal von einem Script erledigt.“

PowerShell Quellen

- Der eigene Rechner (powershell_ise.exe, powershell.exe)
- Suchmaschinen: Das Internet ist voll von Tutorials, Blogs, Tipps , …!
- www.microsoft.com/PowerShell
- Wikipedia PowerShell (de)
- Wikipedia PowerShell (en)
- Frank Koch Windows PowerShell 3.0 (kostenloses eBook)
- PowerShell Scripting Guy, Ed Wilson
- Windows PowerShell Owner's Manual
- …

Was ist PowerShell? (Wikipedia)

- PowerShell ist eine Microsoft Alternative zur Windows Konsole CMD.EXE.
- Den Kern der PowerShell bilden kleine Funktionseinheiten, genannt **Cmdlets** (gesprochen command-lets), die dem Benennungsschema Verb-Substantiv folgen, also beispielsweise **Get-Help** oder **Set-Location**.

  Für die Bezeichnungen einiger PowerShell-Befehle (Cmdlets) sind vordefinierte **Alias-Namen** vergeben, die den Befehlen klassischer Shells entsprechen, unter anderem als Hilfe für **Umsteiger** von der CMD-Kommandozeile.

Was ist PowerShell? (Schwichtenberg)

- Dr. Holger Schwichtenberg
  (Windows PowerShell 4.0 Das Praxisbuch, 2014, 926 Seiten):

> Das DOS-ähnliche Kommandozeilenfenster hat viele Windows-Versionen in beinahe unveränderter Form überlebt. Mit der Windows PowerShell (WPS) besitzt Microsoft nun endlich einen Nachfolger, der es mit den Unix-Shells aufnehmen kann und diese in Hinblick auf Eleganz und Robustheit in einigen Punkten auch überbieten kann. Die PowerShell ist eine Adaption des Konzepts von Unix-Shells auf Windows unter Verwendung des .NET Frameworks und mit Anbindung an die Windows Management Instrumentation (WMI).

Schwichtenberg: „… Umgebung für interaktive Systemadministration …“

In einem Satz: Die Windows PowerShell (WPS) ist eine neue, .NET-basierte Umgebung für interaktive Systemadministration und Scripting auf der Windows-Plattform.

Die Kernfunktionen der PowerShell sind:

- Zahlreiche eingebaute Befehle, die „Commandlets“ genannt werden
- Zugang zu allen Systemobjekten, die durch COM-Bibliotheken, das .NET Framework und die Windows Management Instrumentation (WMI) bereitgestellt werden
- Robuster Datenaustausch zwischen Commandlets durch Pipelines basierend auf typisierten Objekten
- Ein einheitliches Navigationsparadigma für verschiedene Speicher (z. B. Dateisystem, Registrierungsdatenbank, Zertifikatsspeicher, Active Directory und Umgebungsvariablen)
- Eine einfach zu erlernende, aber mächtige Skriptsprache mit wahlweise schwacher oder starker Typisierung
- Ein Sicherheitsmodell, das die Ausführung unerwünschter Skripte unterbindet
- Integrierte Funktionen für Ablaufverfolgung und Debugging
- Die PowerShell kann um eigene Befehle erweitert werden.
- Die PowerShell kann in eigene Anwendungen integriert werden (Hosting).

PowerShell Schlagworte für meinen Einstieg

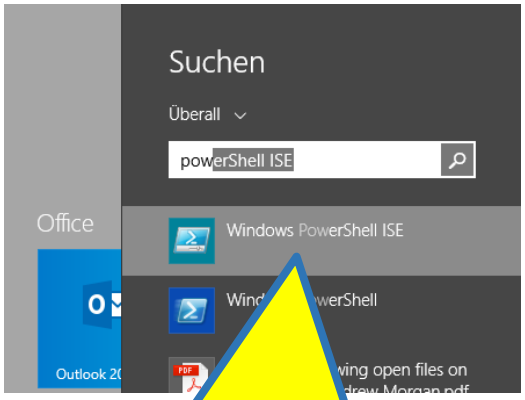
- Commandlets bzw. Cmdlets
- Alias-Namen
- Get-Help
- Objekte
- Pipeline
- einheitliches Navigationsparadigma
- mächtige Skriptsprache
- .Net Framework

… und „System-Administration“?

- PowerShell ist eine „Umgebung für interaktive Systemadministration“
- Ich bin kein Administrator
- Daher: PowerShell-Exploration eines NICHT-Admins
- Viele administrative Fragen, auf die PowerShell sicher eine Antwort hätte, stelle ich gar nicht und kann deswegen auch nichts dazu sagen.
- Meine ersten konkreten PowerShell Ziele sind:
  - Grundfunktionalitäten verstehen (s. o. „Schlagworte“)
  - Kleine alltägliche Registry-Manipulationen „automatisieren“
  - PS-Scripte für meine wenigen Standard-Batches (z.B. xBuild.bat)
  - PS-Script für das Update einer Win-App (s. „Motivation“)

… und … „Umsteiger“? Wo kommen wir her?

- Windows Konsole CMD.EXE
- Befehle, Funktionen:
  cd, cls, copy, del, dir, echo,
  md, move, popd, pushd,
  rd, type, <lw>:?
- Funktionieren diese Befehle in PowerShell?




```
E:
cd \engine\temp
cd \PsTmp
dir
dir *.ps1
```

Hinweis:
Zum Kennenlernen und Entwickeln „PowerShell ISE.exe“ statt „PowerShell.exe“ nutzen. Das bringt deutlich mehr User-Support. „PowerShell.exe“ ist für den operativen Einsatz wichtig (z.B. wg. der Aufrufparameter).
Nebenbei:
„ISE“ steht für ***Interactive Scripting Engine***

Grenzen der Cmd.exe Kompatibilität

```
dir *.
dir *.* /o-d
```

- CMD-Befehle in PowerShell sind offensichtlich nicht die CMD-Original-Befehle.
- Sie sind nicht 1:1 Parameter-Kompatibel.
- CMD-Befehle sind Alias-Benennungen von PowerShell Befehlen, den ***Cmdlets***.

- Welche Cmdlets stecken hinter den Alias-Benennungen?
- Wir kennen aus unseren Schlagworten das Cmdlet „Get-Help“

Cmdlets Vol1

- `Get-Help`

  „Windows PowerShell enthält keine Hilfedateien, Sie können die Hilfethemen jedoch online lesen. Oder laden Sie mit dem Cmdlet "**Update-Help**" Hilfedateien auf den Computer herunter, und zeigen Sie die Hilfethemen anschließend mit dem Cmdlet "Get-Help" in der Befehlszeile an.“
- `Update-Help`
- `Get-Help –Name alias`

  `VERWANDTE LINKS`

  `about_Aliases`

  (hinter about_... steht eine ***PS-Konzept***)
- `Get-Help Get-Alias`

  Gets the aliases for the current session.
- `Get-Alias`

  Liefert die Liste der Cmdlets, die hinter den uns bekannten Aliases stecken.

Hinweis:
Für den Aufruf von „Update-Help“ PowerShell im Admin-Modus starten.

Nebenbei:
Auf diesen Befehl bin ich sehr früh gestoßen. Evtl. hatte ich deswegen nie Probleme mit der PowerShell Hilfe.

Cmdlets Vol2

- `Get-Alias -Name cd`

```
CommandType     Name
-----------     ----
Alias           cd -> Set-Location
```

Mehrere Befehle – z.B. zur besseren Übersicht – in eine Zeile schreiben mit „;“

- `Get-Alias Cd; Get-Alias Dir`
- `cd -> Set-Location`
- `dir -> Get-ChildItem`

Infos zu Set-Location und Get-ChildItem selbstverständlich mit:

- `Get-Help Set-Location`
- `Get-Help Get-ChildItem`
- `Get-Help Set-Location -full`
- `Get-Help Set-Location –examples`

Cmdlets – Common Parameters

Parameter, die jedes Cmdlet unterstützt (in Klammern die Kurzbezeichnungen).

- -Debug (db)
- -ErrorAction (ea)
- -ErrorVariable (ev)
- -OutVariable (ov)
- -OutBuffer (ob)
- -PipelineVariable (pv)
- -Verbose (vb)
- -WarningAction (wa)
- -WarningVariable (wv)

Risiko minimierende “Common Parameters”:

- -WhatIf (wi)
- -Confirm (cf)

Infos dazu mit: `Get-Help about_CommonParameters`

Cmdlets Vol3

Wir wenden unser Wissen an (Cmdlets statt Aliases):

```
"";"Liste";Set-Location e:\engine\ctrl;Get-ChildItem *.exe

Liste

    Verzeichnis: E:\engine\ctrl

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         06.06.2001     02:20      39936 FileDate.exe
-a---         05.05.2011     09:58     121856 HoboCopy.exe
-a---         13.11.2013     22:22    1058199 RawCopy64.exe
-a---         20.11.2010     05:25     128000 RobocopyXP027.exe
```

Cmdlets Vol4

Cmdlets, die uns bekannte Aliase realisieren:

Cd -> Set-Location
Cls -> Clear-Host
Copy -> Copy-Item
Del -> Remove-Item
Dir -> Get-ChildItem
Echo -> Write-Output
Md -> New-Item
Move -> Move-Item
Popd -> Pop-Location
Pushd -> Push-Location
Ren -> Rename-Item
Set -> Set-Variable
Type -> Get-Content

Was sind Cmdlets?

- Cmdlets (gesprochen command-lets) sind die Funktionseinheiten von PowerShell.
- Cmdlets folgen dem Benennungsschema Verb-Substantiv, bspw.
  - **`Get-Help`** oder
  - **`Set-Location`**
  - Liste der „offiziellen Verben“ mit **`Get-Verb`**
- PowerShell 4.0 enthält 538 Cmdlets, 712 Functions und 150 Aliase.
- Cmdlets liefern Objekte zurück.
- Cmdlets können mit der Pipe „|“ verknüpft werden.
- Cmdlets sind als .Net-Klassen implementiert.

Objekte und Rückgabe Objekte

Was liefert Get-ChildItem tatsächlich zurück?

Das Cmdlet, um PS-Objekte zu untersuchen, heißt:

```
Get-Member
```

(Aber Get-Member will ein Objekt haben.)

„Get-Member : Sie müssen ein Objekt für das Cmdlet "Get-Member" angegeben. “

Es lohnt sich die Fehlermeldungen zu lesen

Alles ist ein Objekt

- `Get-Member -InputObject "Michael"`
  `TypeName: System.String`
- `"Michael".Length`
  `7`

Objekt per „Pipe“ an Cmtlet weiterreichen

- `"Michael" | Get-Member`

- `Get-ChildItem | Get-Member`
  `TypeName: System.IO.DirectoryInfo`
  `TypeName: System.IO.FileInfo`

Objects

Was kann man mit Objekten alles machen? Außer Nutzung der eigenen Methoden.

```
get-help object
```

liefert Liste spezieller Object-Cmdlets (und es gibt noch weitere Cmdlets).

Name	Category	Synopsis
ForEach-Object	Cmdlet	Performs an operation against each item in a c...
Where-Object	Cmdlet	Selects objects from a collection based on the...
Compare-Object	Cmdlet	Compares two sets of objects.
Group-Object	Cmdlet	Groups objects that contain the same value for...
Measure-Object	Cmdlet	Calculates the numeric properties of objects, ...
New-Object	Cmdlet	Creates an instance of a Microsoft .NET Framew...
Register-ObjectEvent	Cmdlet	Subscribes to the events that are generated by...
Select-Object	Cmdlet	Selects objects or object properties.
Sort-Object	Cmdlet	Sorts objects by property values.
Tee-Object	Cmdlet	Saves command output in a file or variable and...
Get-WmiObject	Cmdlet	Ruft Instanzen von Klassen der Windows-Verwalt...
Remove-WmiObject	Cmdlet	Löscht eine Instanz einer vorhandenen Klasse d...
about_Objects	HelpFile	Provides essential information about objects i...
about_Object_Creation	HelpFile	Explains how to create objects in Windows Powe...

Objects und Pipeline

Welche Werte liefert das Property „Attributes“ des Get-ChildItem-Objekts?

```
Get-ChildItem | ForEach-Object {$_.attributes}
Directory
Archive
```

`|`	Objekt „weiterleiten“ zum nächsten Befehl
`{}`	ist die Syntax für einen Codeblock.
`$_`	ist die Syntax für den Zugriff auf ein Item eines Objects.
`attributes`	das Property findet man mit Get-Item|Get-Member

Nebenbei, es gibt viele Abkürzungen. Beispiel:

```
gci | ForEach {$_.attributes}
```

`$_` ist die Abkürzung für **`$PSItem`**

Objects und Pipeline

An **„`Dir *.`“** waren wir eben gescheitert. Nachbildung mit **`Get-ChildItem`**:

Objekt „weiterleiten“ zum nächsten Befehl mit Pipe „|“

```
Get-ChildItem | Where-Object {$_.attributes -match "Directory"}
```

`|`	Objekt „weiterleiten“ zum nächsten Befehl
`{}`	ist die Syntax für einen Codeblock.
`$_`	ist die Syntax für den Zugriff auf ein Item eines Objects.
`attributes`	das Property findet man mit Get-ChildItem|Get-Member
`-match`	den Parameter findet man mit Get-Help|Where-Object
`"Directory"`	mit „Get-ChildItem|ForEach-Object {$_.attributes}“ ermittelt

Cmdlets Vol5 „alles ist in Laufwerken“

```
Get-PSDrive
```

- `Get-ChildItem alias:c*`
- `Get-ChildItem Env:`
- `Get-ChildItem Env:path`
- `(Get-ChildItem Env:path).value`
- …

Zugriffe auf alles:
Files, Environment, Registry (HKLM u. HKCU), Cert, Functions, Variablen, …

Alles ist ein „Laufwerk“!
Das ist wohl das

„einheitliche Navigationsparadigma“

von Schwichtenberg s.o.

Dateien anzeigen, öffnen

```
Cd Script
Get-Content -Path Greet-World.ps1
Cd ..
Get-Content x.csv
Get-Content $env:userprofile\ineta.xls*
Invoke-Item $env:userprofile\ineta.xls*
```

Der komplette Pfad (oder „.\“, falls die Datei im akt. Verz. steht) wird für den ausführenden Aufruf benötigt, das erfordert das PS-Sicherheitskonzept. Es sein denn, die ausführbare Datei steht im Suchpfad.

```
Start-Process -FilePath notepad -ArgumentList greet-world.ps1
Get-Process -Name notepad
Stop-Process -Name notepad -WhatIf
Stop-Process -Name notepad
Stop-Process -Name notepad
Stop-Process -Name notepad -ErrorAction SilentlyContinue
```

Vor der Skriptverarbeitung nebenbei noch folgende Frage: Welche PowerShell Version läuft eigentlich gerade?

- `Get-Host`
- `$PSVersionTable` `# scheint richtiger`

Scripte – about Execution Policies

PowerShell_ISE.exe

- starten und erläutern
- Ansicht -> Symbolleiste, Skriptbereich, Befehls Add-On
- "HalloWelt" # Ausführen liefert:
  Hallo Welt
- Als Script Unbenannt1.ps1 speichern und ausführen, liefert:

```
PS E:\PsTmp> E:\PsTmp\Unbenannt1.ps1
Die Datei "E:\PsTmp\Unbenannt1.ps1" kann nicht geladen werden, da die Ausführung von
Skripts auf diesem System deaktiviert ist. Weitere Informationen finden Sie unter
"about_Execution_Policies" (http://go.microsoft.com/fwlink/?LinkID=135170).
    + CategoryInfo          : Sicherheitsfehler: (:) [], ParentContainsErrorRecordExcept
   ion
    + FullyQualifiedErrorId : UnauthorizedAccess
```

- On the fly in zweitem PS-Fenster im Admin-Modus:
  `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned`

Damit ist die Skriptverarbeitung freigeschaltet.

Scripte – Hallo Welt!

Infos (Write-HalloWelt.ps1)

```
# #               Inline Kommentar
# <#  ...  #>     Blockkommentar
# `               Code-Zeilen-Umbruch
# ;               Befehle in einer Zeile trennen
# (...)           Runde Klammern haben algebraische Bedeutung:
#                 Sie teilen der Shell mit, was zuerst ausgeführt wird.
# $               Variablen-Kennzeichen
[String]$HalloWelt = "Hallo Welt"     # Typisierung ist nicht zwingend
Write-Host ""; Write-Host $HalloWelt  # in ISE sind die Var. nach Ausf. <F5> noch da
Write-Host "mit         ", ("PowerShell-Version" + " " + (Get-Host).Version.ToString() )
Write-Host "auf PC      ", (Get-Content Env:Computername)
Write-Host "mit CPU-Typ ", (`
 Get-ItemProperty -path `
 "HKLM:\SYSTEM\CurrentControlSet\Control\Session Manager\Environment"`
 ).PROCESSOR_ARCHITECTURE
<#
 Write-Host "mit CPU-Typ ", (Get-Content Env:Processor_Architecture)
#>
Write-Host "von User    ", (Get-Content Env:Username)
```

Hinweis:
Störende Variablen mit „Remove-Variable“ löschen.

Ende von Teil 1 – PowerShell Konzepte

Material zu dieser Session unter

https://github.com/MichaelPae/PS-Pieces

Vielen Dank für Eure Aufmerksamkeit!