Aufgaben für den zweiten Tag

Auf den folgenden Seiten werden kleine Aufgaben zusammengestellt, die jeder Teilnehmer am Ende des zweiten Schulungstages innerhalb von 45 bis 60 Minuten umsetzen soll.

Letzte Aktualisierung: 05/09/2020

Wichtig: Im Unterschied zu den Aufgaben des ersten Tages sind die Aufgaben des zweiten Tages bereits etwas anspruchsvoller. Aber auch hier gilt, wer zu einer Aufgabe keine Idee hat, bitte die Aufgabe einfach auslassen oder, noch besser, den Dozenten fragen. Die meisten Aufgaben kamen in ähnlicher oder identischer Form am heutigen Tag vor.

Wichtig: Einige Aufgaben verwenden Commands aus dem Modul **PoshKurs**, das am ersten Tag der Schulung hinzugefügt wurde (wenn nicht, bitte den Dozenten fragen). Das Modul ist auch Teil der Übungsbeispiele und kann per **Import-Module** aus dem Verzeichnis geladen werden.

Aufgabe 1: Hotfix-Abfrage über eine Datei mit Hotfix-Ids

Ausgangspunkt ist eine Datei *Hotfixliste.txt* (liegt als separate Datei im Material-Verzeichnis vor), die eine Reihe von Hotfix-Ids enthält. z.B. KB4514359, KB4493478 usw.

Der Inhalt der Datei soll mit einem Skript eingelesen und jede Hotfix-ID soll ausgegeben werden.

Damit eine sinnvolle Ausgabe entsteht, müssen einzelne IDs eventuell geändert werden.

Wie kann die Überschrift „Hotfix“ ausgelassen werden?

Aufgabe 2: Prüfen, ob ein Hotfix installiert ist

Für jede aus der Datei *Hotfixliste.txt* eingelesene Hotfix-ID soll per **Get-Hotfix**-Cmdlet geprüft werden, ob der Hotfix installiert ist

Damit keine Fehlermeldung angezeigt wird, wird der Parameter *–ErrorAction* mit dem Wert *Ignore* gesetzt.

Dann muss per $? abgefragt werden, ob der Befehl erfolgreich ausgeführt oder nicht.

Aufgabe 3: Ausgabe nicht vorhandener Hotfixes

Wie die letzte Aufgabe, es sollen aber nur die Hotfixes ausgegeben werden, die nicht vorhanden sind.

Aufgabe 4: Ausgabe in eine Datei

Wie die letzte Aufgabe, nur sollen die Namen der Hotfixes, die nicht vorhanden sind, in eine Datei geschrieben werden.

Erweiterung: **Get-Hotfix** besitzt leider keinen Computername-Parameter. Per PowerShell-Remoting kann die Abfrage trotzdem auch gegen andere Computer im Netzwerk durchgeführt werden. Eine Alternative ist WMI über das **Get-CIMInstance**-Cmdlet mit der

Klasse *Win32_QuickFixEngineering* im Rahmen einer zuvor angelegten CIM-Session. Das Thema ist eventuell erst am dritten Tag an der Reihe.

Eine Remote-Abfrage muss natürlich technisch möglich sein. Ansonsten wird Localhost bzw. der Rechnername verwendet.

Aufgabe 5: Ausgabe von Computernamen

Der Befehl **Get-Computerkonten** aus dem Modul **PoshKurs** gibt ("virtuelle", d.h. nicht existierende) Computerkonten zurück.

a) Gesucht ist ein Befehl, der nur Computerkonten für Namen ausgibt, die mit "Server" beginnen.

b) Gesucht ist ein Befehl, der nur Computerkonten für Namen ausgibt, die nicht mit "Server" beginnen.

c) Gesucht ist ein Befehl, der nur Computerkonten für Namen ausgibt, die einer 0 oder 2 enden.

d) Gesucht ist ein Befehl, der nur Computerkonten ausgibt, die vor 7 Uhr das letzte Mal gebootet wurden.

e) Gesucht ist ein Befehl oder eine Befehlsfolge, die angibt, ob ein bestimmter Name, der per **Read-Host** zuvor eingegeben wurde, existiert.

Aufgabe 6: Ausgabe der installierten Anwendungen

Der Befehl **Get-Uninstallapp** gibt die Eckdaten zu allen installierten Anwendungen zurück (auf der Grundlage der Registry).

a) Gesucht ist ein Befehl, der nur die Namen der Anwendungen in eine Datei schreibt (Umleitung per >)

b) Gesucht ist ein Befehl, der die Ausgabe nach dem Namen der Anwendungen sortiert.

c) Gesucht ist ein Befehl, der nur Anwendungen ausgibt, für die es eine Versionsnummer gibt.

d) Gesucht ist ein Befehl, der die Anwendungen nach ihrem Verzeichnis gruppiert (Group-Object)

Aufgabe 7: Auslesen von Computernamen aus dem AD

Das Command **Get-AdComputer** holt Computerkonten aus dem AD.

Gesucht ist ein Befehl, der nur die Namen der Computerkonten in eine Textdatei mit dem Namen *Computerkonten.txt* schreibt. Diese Datei gibt es am Anfang noch nicht.

Hinweis: Steht kein AD zur Verfügung, kann alternativ das Command **Get-Computerkonten** aus dem PoshKurs-Modul verwendet werden. Dieses Command "simuliert" eine AD-Abfrage und liefert ebenfalls Objekte zurück, die aber nicht auf real existierenden Konten basieren.

Aufgabe 8: Erreichbarkeitsabfrage

Erstelle ein kleines Skript (Ps1-Datei) mit PowerShell ISE oder Visual Studio Code, das per Get-Content den Inhalt der Textdatei Computerkonten.txt ausliest und für jeden Computer eine Erreichbarkeitsabfrage mit Hilfe des Command Test-Connection durchführt.

Hinweis: Steht kein LAN zur Verfügung oder wurde in der letzten Übung das Command Get-Computerkonten ausgeführt, kann anstelle von Test-Connection der Befehl Test-Ping aus dem PoshKurs-Modul verwendet werden. Dieser Befehl simuliert ebenfalls etwas, in diesem Fall eine Ping-Abfrage. Es werden daher Zufallsergebnisse zurückgegeben.

Da es bei der Aufgabe nur um die Umsetzung geht, spielt dieser Umstand keine Rolle.

Aufgabe 9: Erreichbarkeitsabfrage mit Report

Das Skript aus dem letzten Beispiel soll dahingehend erweitert werden, das die Namen aller nicht erreichbaren Computer in eine Textdatei geschrieben wird.

Aufgabe 10: Ausgabe einer kleinen Statistik als Zusammenfassung

Das Skript aus dem letzten Beispiel soll dahingehend erweitert werden, dass am Ende eine kleine "Statistik" mit der Anzahl der erreichbaren und der nicht erreichbaren Computer ausgegeben werden.

Welche Form der Ausgabe eignet sich am besten, wenn zwei Zahlen zusammen mit Text in einer Ausgabe kombiniert werden sollen?

Beispiel:

Erreichbar	Nicht erreichbar
10	2

Wie wird wird eine solche Ausgabe erreicht?

Aufgabe 11: Erreichbarkeitsabfrage mit Steuerung

Die Textdatei *Computerkonten2.txt* besteht aus zwei Spalten. Die erste Spalte enthält die Namen von Computern im lokalen Netzwerk.

Hinweis: Aktuell sind nur Standardnamen eingetragen. Trage hier ein paar Namen aus dem echten LAN ein.

Die zweite Spalte enthält entweder ein Ja oder ein Nein.

Ja bedeutet, dass der Computer abgefragt wird, Nein bedeutet, dass er ausgelassen ist.

Ändere den Befehl für die Abfrage der Computer so ab, dass nur Computer abgefragt werden, bei denen in der zweiten Spalte ein "Ja" steht.

Aufgabe 12: Inspizieren der Rückgabe einer Erreichbarkeitsabfrage

Der Aufruf von Test-Connection gibt immer ein Objekt zurück, das wie jedes Objekt zahlreich Members besitzt.

Wie lassen sich die Properties von dem Objekt ausgeben, das von **Test-Connection** zurückgegeben wird?

Tipp: Get-Member besitzt u.a. einen **MemberType**-Parameter.

*** Ende der Aufgaben für den 2. Tag ***