Aufgaben für den dritten Tag

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

Letzte Aktualisierung: 07/09/2020

Aufgabe 1: Erreichbarkeitstest mit Textdatei (Level A)

Ausgangspunkt ist eine Textdatei *Computernamen.txt*, die pro Zeile den NetBIOS-Namen eines Computers im Netzwerk erhält.

Diese Datei muss entweder angelegt werden oder wurde im Rahmen der Übung zur Verfügung gestellt.

1) Jeden Computer ansprechen

Gesucht ist eine Befehlsfolge, die pro Name die Erreichbarkeit des Computers per Test-Connection-Cmdlet testet.

Hinweis: Das **Test-Connection**-Cmdlet soll mit den Parametern –Count 1 und –Quiet aufgerufen werden, so dass pro Computer entweder ein True oder ein False-Wert zurückgegeben wird.

2) Erweitern der Ausgabe

Erweitern Sie den Befehl aus der letzten Aufgabe so, dass nur die Namen aller erreichbaren Computer ausgegeben werden. Die Computer, die nicht erreichbar sind, werden nicht ausgegeben.

Welche Form der Ausgabe eignet sich am besten?

3) Namen in einer Datei speichern

Wie werden die Namen in einer Variablen gesammelt, so dass die Namen am Ende z.B. in eine Textdatei geschrieben werden können?

4) Ausgabe einer Statistik

Am Ende sollen auch 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 in einer Ausgabe kombiniert werden sollen?

Aufgabe 2: Erreichbarkeitstest mit Textdatei, die aus mehreren Spalten besteht (Level A)

Die Aufgabe entspricht grundsätzlich der letzten Aufgabe. Bitte aber in einer anderen Ps1-Datei speichern.

Ausgangspunkt ist eine Textdatei Computernamen.txt, die dieses Mal pro Zeile zwei Spalten besitzt:

Den NetBIOS-Namen eines Computers im Netzwerk und entweder eine 1 oder eine 0.

Dieses Mal soll der betreffende Computer nur dann angepingt werden, wenn der Wert der zweiten Spalte 1 ist. Am Ende soll in der Statistik auch die Anzahl der ausgelassenen Computer enthalten sein.

Aufgabe 3: Erreichbarkeitstest mit einer Function (Level B)

Diese Übung setzt voraus, dass das Prinzip von Functions ganz grob bekannt ist.

Ausgangspunkt ist eine Datei Computernamen.txt, die pro Zeile den NetBIOS-Namen von Computern im Netzwerk enthält.

Erstellen Sie eine mit dem function-Befehl eine Function, die einen Namen als Parameter erwartet und die prüft, ob der Name in der Datei enthalten ist. Die Function soll die Werte $true oder $false zurückgeben.

Erweitern Sie die Function, so dass auch der Pfad der Datei als Parameter übergeben werden kann.

Erweitern Sie die Function, so dass zu Beginn geprüft wird, ob die Datei existiert.

Aufgabe 4: Erreichbarkeitstest über ein Skript (Level B)

Bauen Sie die Function aus dem letzten Beispiel in ein Skript ein. Das Skript soll mit zwei Parametern aufgerufen werden: Der Pfad der Textdatei mit den Computernamen und den Namen des gesuchten Computers. Innerhalb des Skripts wird die Function aufgerufen.

Die Parameter werden zu Beginn des Skripts mit dem **param**()-Befehl angegeben.

Beispiel: param($Pfad, $Computername)

Aufgabe 5: Auslagern der Function in eine eigene Ps1-Datei (Level B)

Lagern Sie die Function aus dem letzten Beispiel in eine Ps1-Datei aus. Die Ps1-Datei soll von dem Skript, das prüft, ob ein Name in einer Datei enthalten ist, so ausgeführt werden, dass die Function anschließend aufgerufen werden kann.

Erweitern Sie die Function, so dass auch ein Ähnlichkeitsvergleich möglich ist. Zum Beispiel findet der Suchname „LINT“ alle Computernamen, die mit „LINT“ beginnen.

Aufgabe 6: Umwandeln einer Ps1-Datei in eine Psm1-Datei (Level B)

Machen Sie aus dem Skript mit der Function ein Modul, in dem Sie die Ps1-Datei in eine Psm1-Datei umbenennen, ein Verzeichnis mit dem Namen der Datei in einem Modulverzeichnis (z.B. C:\Program Files\WindowsPowerShell\Modules) anlegen und die Psm1-Datei in dieses Verzeichnis kopieren. Von jetzt an sollte die Function automatisch nach jedem PowerShell-Start zur Verfügung stehen.

Tipp: Die Pfade aller in Frage kommenden Modulverzeichnisse sind in der Umgebungsvariablen PSModulePath enthalten.

Das Modul heißt so wie das Verzeichnis heißt (z.B. Pskurs – die Psm1-Datei heißt dann Pskurs.psm1).

Frage: Wie wird das neue Modul per Import-Module geladen?

Frage: Kann eine Psm1-Datei direkt ausgeführt werden?

Aufgabe 7: Zeitpunkt des letzten Bootens eines Computers im Netzwerk per WMI abfragen (Level B)

Ausgangspunkt ist eine Function mit dem Namen **Get-Computer**, die die Namen von Computern im Netzwerk zurückgibt – z.B. durch Einlesen aus der Textdatei.

```
function Get-Computer
{
  Get-Content –Path .\Computernamen.txt
}
```

Eine zweite Function **Get-LastBootTime** erwartet den Namen eines Computers und gibt per WMI (Get-CIMInstance Win32_OperatingSystem) den Zeitpunkt des letzten Bootens zurück.

```
function Get-LastbootTime
{
  param($Computername)
  ??? hier fehlt noch etwas ???
}
```

Jetzt kommt die Aufgabe: Wie werden die beiden Functions im Rahmen einer ForEach-Object-Wiederholung kombiniert, dass von jedem Computer, sofern er erreichbar ist, der Zeitpunkt des letzten Bootens ausgegeben wird?

Erweiterung für Fortgeschrittene: Erstellen Sie die Function Get-LastBootTime so, dass sie den Namen des Computers auch aus der Pipeline holen kann, aber nicht muss. Außerdem sollen beide Functions eine Hilfe enthalten und die Parameter einen Datentyp besitzt.

Das Ergebnis soll als Html mit einer Tabelle ausgegeben werden.

Aufgabe 8: Umbenennen von Verzeichnissen (Teil 1)

Gegeben sei ein Verzeichnisinhalt, in dem bereits eine Reihe von Unterverzeichnissen enthalten sind: PS01,PS02,PS03,PS04

Der folgende Befehl legt z.B. 10 Verzeichnisse an:

```
(1..10).foreach{mkdir ("PS{0:00}" –f $_)
```

Sollten die Verzeichnisse bereits existieren, sind natürlich entsprechende Fehlermeldungen die Folge.

Alle Verzeichnisse sollen mit einem einzigen Befehl in PowerShell_01, PowerShell_02 usw. umbenannt werden.

Aufgabe 9: Umbenennen von Verzeichnissen (Teil 2)

Im ersten Teil der Übung sollen 12 Verzeichnisse mit dem Namen Jan bis Dez mit einem einzigen Befehl angelegt werden.

Tipp: Ein Get-Date –Month 1 –Format MMM liefert z.B. „Jan“

Im zweiten Teil der Übung sollen alle Dateien aus einem beliebigen Verzeichnis, z.B. dem Documents-Verzeichnis in Abhängigkeit ihres Datums, an dem sie angelegt wurden (CreationTime-Eigenschaft) in den entsprechenden Ordner kopiert werden. Eine Datei, die im Januar angelegt wurde (Jahr und Tag spielen keine Rolle), soll in den Ordner „Jan“ kopiert werden, eine Datei, die im Februar angelegt wurde, entsprechend auf den Ordner „Feb“ usw.

Tipp: Ein (Get-Item –Path C:\Windows\Win.ini).CreationTime.ToString("MMM") liefert den Monat, an dem die Datei Win.ini angelegt wurde, z.B. als „Okt“.

Aufgabe 10: Zugriff auf die Registry (Level A)

Das Ziel dieser Übung ist eine Vertrautheit im Umgang mit den Item-Cmdlets Get-ChildItem, Get-Item und Get-ItemProperty und ein Verständnis für die Weiterverarbeitung der Rückgabewerte.

Zugriff auf die Registry am Beispiel von ODBC-Datenquellen

1) Auflisten aller System-DSNs

Ein System-DNS ist ein Eintrag in der Registry (oder als Datei), über den eine Datenquelle (z.B. SQL Server-Datenbank) angesprochen wird.

Doch wie soll das gehen? Ganz einfach, über das dir-Kommando (also Get-ChildItem), auf das der Pfad des Schlüssels folgt. Dieser lautet HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI.

Hinweis: Der Hive-Name „HKEY_LOCAL_MACHINE“ muss bei der PowerShell durch den Namen des Laufwerks (in diesem Fall „HKLM:“) ersetzt werden.

Gab die Abfrage nichts zurück? In diesem Fall gibt es keine System-DSNs auf diesem System. Mit hoher Wahrscheinlichkeit gibt es aber Benutzer-DSNs. Sie werden in der Registry unter dem Schlüssel HKEY_CURRENT_USER\SOFTWARE\ODBC\ODBC.INI

Wie heißt der PowerShell-Befehl, der die Unterschlüssel dieses Schlüssels auflistet?

Jetzt kommt ein Ergebnis zurück.

Jeder Schlüssel kann eine Reihe von Einträgen besitzen. Bei einem DSN ist dies z.B. der Name des Treibers (Driver). Bei der PowerShell werden die Einträge als Properties des Item-Objekts geführt, das den Schlüssel repräsentiert.

Wenn Get-ChildItem bzw. Get-Item einen Schlüssel holt, wie heißt dann das Cmdlet, das eine Property dieses Items holt?

Lösung: Get-ItemProperty

Wie heißt der Befehl, der zu jedem Benutzer-DSN den Namen des Treibers (und nur diesen) zurückgibt?

2) Export einer Rückgabe in eine Textdatei

Exportieren Sie die Namen aller Benutzer-DSN-Treiber in eine Textdatei – diese soll am Ende nur die Namen enthalten (und sonst nichts).

Aufgabe 11: Umwandeln einer Befehlsfolge in ein Skript

Das Ziel dieserAufgabe ist es, eine relativ einfache Abfrage, die bislang durch Eingabe in die Konsole ausgeführt wird, per Skript umsetzen, so dass die Befehlsfolge a) nicht immer wieder eingegeben und b) dank Parametern auch flexibler ausgeführt werden kann.

1) Ausgangspunkt ist der folgende Befehl:

Get-EventLog –Logname Security –InstanceId 4624 –Newest 20

Dieser Befehl soll in der ISE eingefügt und als Datei mit dem Namen „SecEventAbfr.ps1“ gespeichert werden. Damit wurde aus einem Befehl ein Skript.

Wichtig: Mit dem Speichern als Ps1-Datei greift bei der Ausführung die Ausführungsrichtlinie, so dass es passieren kann, dass anstelle eines Ergebnisses eine Fehlermeldung die Folge ist.

2) Wie wird das Skript in der Konsole ausgeführt?

---

3) Erweitern um einen Kommentarblock

Das Skript soll mit einem mehrzeiligen Kommentarblock erweitert werden. Dieser soll ein .Synopsis enthalten, auf das in der nächsten Zeile eine kurze Beschreibung folgt.

Aufgabe 12: Erweitern eines Skripts mit einem Parameter

Mit diesem Schritt kommen Parameter ins Spiel, über die das Skript flexibler ausgeführt wird.

1) Identifizieren der variablen Werte

In dem Abfragebefehl gibt es zwei variable Werte: Die Instance Id und die Anzahl der zurückgegebenen Einträge. Beide sollen durch Variablen ersetzt werden

2) Umwandeln von Variablen in Parameter

Die Variablen *$EventId* und *$AnzahlEvents* werden mit dem **param**-Befehl zu Parametern des Skripts

3) Ausführen des Skripts mit Argumenten (Parameterwerten)

Wie wird die Ps1-Datei mit den Parametern ausgeführt, d.h. wie werden die Werte (Argumente) für die Parameter EventId und AnzahlEvents beim Aufruf festgelegt?

---

4) Die Rolle der Paramternamen

Können die Namen der Parameter weggelassen werden? Wenn ja, warum?

---

5) Festlegen eines Default-Wertes für einen Parameter

Der Parameter **AnzahlEvents** soll mit dem Default-Wert 20 belegt werden, so dass für diesen Parameter kein Wert übergeben werden muss.

Aufgabe 13: Hinzufügen eines Switch-Parameters

In dieser Übung wird zu den Parmetern des Skriptes ein switch-Parameter hinzugefügt. Ein Switch-Parameter wird nur angegeben, auf ihn folgt kein Wert.

1) Hinzufügen eines Switch-Parameters

Das Skript soll um den Switch-Parameter HTMLReport erweitert werden.

Tipp: Ein Switch-Parameter besitzt den (PowerShell-) Datentyp Switch.

2) Abfragen, ob ein switch-Parameter angegeben wurde

Wie wird abgefragt, ob der Switch-Parameter beim Aufruf angegeben wurde?

---

3) Ausgabe der Ausgabe im HTML-Format

Wurde der Parameter HTMLReport übergeben, soll die Ausgabe im HTML-Format ausgegeben werden.

Tipp: Eventuell ist es eine Idee, das Ergebnis der Abfrage zuerst einer Variablen zuzuweisen.

4) Wie sieht die Hilfe des Skripts jetzt aus nachdem mehrere Parameter definiert wurden?

Aufgabe 14: Erweitern der Hilfe

In dieser Übung soll die bereits vorhandene Kommentarblock-Hilfe des Skriptes um Beispiele erweitert werdne.

Beispiele folgen im Kommentarblock auf das Schlüsselwort .Example

1) Die Hilfe zu dem Skript soll um zwei Beispiele erweitert werden.

---

2) Wie muss die Hilfe aufgerufen werden, damit die Beispiele angezeigt werden?

Aufgabe 15: Prüfen, ob das Skript als Administrator gestartet wurde

Da die Abfrage des Security-Eventlogs nur ausgeführt werden kann, wenn das Skript als Administrator gestartet wurde, soll dies vor dem Ausführen der Abfrage geprüft werden. Dafür gibt es die Direktive **#requires**.

1) Wie genau muss **#requires** an den Anfang gestellt werden, damit die Abfrage nicht ausgeführt wird, wenn das Skript nicht als Administrator gestartet wurde.

*** Ende der Aufgaben für den 3. Tag ***