הקדמה

Handbook זה מכיל סקירה של ה Syntax של Windows PowerShell בחלקו הראשון, ובחלקו השני עקרונות לפיתוח command-lets.
ה handbook נכתב ע"י **שחר גבירץ**, ותואם ל Windows PowerShell גרסא 1.0.

כתובת הדוא"ל שלי, אליה ניתן להפנות שאלות והערות, היא Shahar@IShahar.net.
כתובת הבלוג שלי, בו אני כותב בין השאר גם על Windows PowerShell:
http://blogs.microsoft.co.il/blogs/shahar

Syntax

הערות

הערות מסומנות באמצעות הסימן #:

#This is a comment :-)

PowerShell אינו תלוי רישיות

הפקודות והסקריפטים ב Windows Power Shell אינם רגישים להבדל בין אותיות גדולות לקטנות.

שימוש ב Command Lets

השימוש ב cmdlets, הפקודות ב Windows PowerShell, פשוט עד מאד. ראשית, צריך לזכור את המבנה: verb-noun. כל ה cmdlets יהיו במבנה הזה. השתדלו, גם בפונקציות שלכם לתת שמות בתבנית של verb-noun. השימוש יהיה בתבנית הבאה:

Cmdlet position0 position1 –ParameterName value –ParameterSecond value

מיד אחרי שם ה cmdlet, יופיעו פרמטרים מבוססי position, במידה וישנם. אחריהם, או במידה ואין יופיעו פרמטרים שנקראים לפי שם הפרמטר. קודם מופיע שם הפרמטר עם סימן מקף לפניו, ולאחר מכן הערך. במידה ומדובר ב switch, יופיע רק שם ה switch כשלפניו סימן מקף. למשל בדוגמא הבאה, אנחנו מעבירים את d* שהוא wildcard pattern בתור פרמטר ב position 0 ולאחריו משתמשים ב recurse switch– שמציג לנו בצורה רקורסיבית את כל הקבצים שמתחילים באות d במבנה התיקיות :

get-childitem d* –recurse

אנחנו יכולים להעביר את הערך d* גם לפי שם הפרמטר בצורה הבאה:

get-childitem -filter d* -recurse

כדי לקבל פרטים על cmdlet מסויים, יש להשתמש ב get-command:

Get-command get-childitem

בצורה זאת, אנחנו נקבל פרטים על ה cmdlet ששמו get-childitem.

©כל הזכויות שמורות ל **שחר גבירץ**.

הצהרה על משתנה

הצהרה על משתנה מתבצעת ללא ציון הסוג (Type) – הסוג נקבע לפי ההערך שמושם. לפני שם המשתנה יש לשים את סימן הדולר:

```
$myInt = 123
$myFloat = 12.7
$myString = "Hello World!"
```

בדוגמא הבאה, נכניס את התאריך לתוך משתנה. למעשה, מה cmdlet (עיין ערך) get-date חוזר לנו אובייקט System.DateTime:

```
$myDate = get-date
```

ניתן גם להצהיר על משתנה בדרך הבאה:

```
[int]$num
[string]$str
```

קבועים

במידה וצריך ליצור קבוע, ניתן להשתמש במשתנה set-variable בצורה הבאה:

```
Set-Variable –name PI –value 3.142 –option constant
```

קריאתו תתבצע באמצעות $PI

מערכים

הגדרת מערך תתבצע בצורה הזאת:

```
$a = “hello”,”my”,”lovely”,”array”
```

גישה לערכי מערך תתבצע בצורה הזאת:

```
PS>$b = $a[2]
PS>$b
PS>lovely
```

שימוש באופרטורים חשבוניים

```
80GB/700MB
>117.028571428571
25/5
>5
"Hello " + "World " + 7 + 5
>Hello World 75
```

יצירת אובייקטים

יצירת אובייקט net. :

```
$myObj = new-object System.Net.WebClient
```

יצירת אובייקט COM:

```
$myComObj = new-object  –com  Sapi.SpVoice
```

סימנים ואופרטורים לוגיים

ב Windows Power Shell ישנם דרכים חדשות לייצוג אופרטורים וסימנים שונים (שימו לב, שהאופרטורים מתחילים במקף):

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

האופרטור\ערך	המשמעות
-eq	שוויון
-ne	לא שווה
-lt	קטן מ-
-gt	גדול מ-
-ge	גדול מ- או שווה ל-
-le	קטן מ- או שווה ל-
-like	השוואה באמצעות wildcard. למשל: Get-Process | Where{$_.company – like “micro*”}
-notlike	שלילה באמצעות wildcard, ההיפך של like
-clike	השוואה באמצעות wildcard, בדומה לlike , רק עם רגישות לאותיות רישיות (case sensitive)
-cnotlike	שלילה באמצעות wildcard עם רגישות לאותיות רישיות. ההיפך מ clike
-match	בדיקת התאמה באמצעות Regular Expression
-notmatch	שלילת התאמה באמצעות Regular Expression, ההיפך מ match
`	סימן Back Tick משמש לשבירת שורה. נמצא ברוב המקלדות הסטנדרטיות בתצורת הקלדה אנגלית QWERTY משמאל למספר 1.

Switches קבועים

ה switches הבאים, בעלי משמעות בכל הפקודות ב Windows Power Shell **שמממשות אותן:**

-confirm : מבקש אישור לפעולה. למשל:

```
get-process sql* | stop-process –confirm
```

-whatif : מדגים מה יקרה במקרה של הרצת הפקודה.

```
get-process sql* | stop-process –whatif
```

-verbose : מתעד את הפעולות שבוצעו.

```
get-process sql* | stop-process –verbose
```

-debug : מציג מידע נוסף אף על המידע שמוצג ב Verbose.

```
Get-process sql* | stop-process -debug
```

קריאה למתודה

קריאה למתודה סטטית תתבצע לפי התבנית הבאה:

<ClassName>::<MethodName>(<parameter>)

לדוגמא:

```
[Console]::WriteLine("Hello {0} World!", "beautiful")
>Hello beautiful World!
```

קריאה למתודה שאינה סטטית, תתבצע לפי התבנית הזאת (פרמטרים יופרדו בפסיק):

$instanceName.MethodName(parameters,parameter,parameter)

למשל, הנה קטע קוד:

```
$myComObj = new-object  –com  Sapi.SpVoice
$myComObj.Speak("Hello, I am Text to Speech")
```

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

בשורה השנייה אנו קוראים למתודה Speak ומעבירים אליה פרמטר מסוג מחרוזת.

שימוש ב command lets

השימוש הבסיסי ביותר יהיה באמצעות כתיבת שם ה cmdlet או Alias שלו. למשל, הדוגמא הבאה תדפיס את כל ה processes הפועלים במערכת:

```
Get-process
```

ניתן גם להעביר פרמטרים. הדוגמא הבאה תחזיר רק פרוססים ששמם מתחיל ב sql:

```
Get-process sql*
```

שימוש ב cmdlets עם סימן ה Pipe

סימן ה pipe (|) משמש לשרשור פקודות אחת לשנייה, המעבירות את המידע בינהן בצורת אובייקטים. לדוגמא:

```
Get-Process | Select-Object name,company
```

יחזיר את ה-Processes, כאשר עבור כל Process יחזיר את שמו ואת שם החברה המייצרת.

הדוגמא הבאה, תחזיר לנו את ה members באובייקט מסויים שנעביר, ואת סוג האובייקט:

```
$datetime = get-date
$datetime | get-member
```

בשורה השנייה, אנחנו מבקשים לקבל את ה members ב instance ששמו datetime.

Output Redirection

ב Windows PowerShell ישנם אופרטורים להפניית פלט. על-מנת להפנות פלט, יש להשתמש בסימן < (גדול מ-). למשל:

```
Dir > $null
```

לא ידפיס שום פלט, מאחר שההפנייה מתבצעת למקום ריק. לעומת זאת:

```
Dir>c:\file.txt
```

תפנה את הפלט לקובץ c:\file.txt, במידה ואינו קיים היא תיצור אותו ובמידה והוא קיים ויש בו מידע, היא תדרוס את כל הנתונים הקיימים. במידה ולא רוצים שהנתונים יידרסו יש להשתמש באופרטור << (2 סימני גדול מ-):

```
Dir >> c:\file.txt
```

ExecutionPolicy

ב Windows PowerShell ישנה אפשרות להגדיר רמות אבטחה שונות להרצת סקריפטים. כברירת מחדל, ההגדרה היא Restricted שתמנע הרצת סקריפטים. ניתן לראות מה ההגדרה באמצעות Get-ExecutionPolicy מומלץ לשנות ל RemoteSigned בצורה הבאה:

```
Set-ExecutionPolicy RemoteSigned
```

ישנן רמות שונות של הגדרת ריצה:

Restricted (ברירת מחדל)

מאפשר הרצת פקודות יחידות, לא מאפשר הרצת סקריפטים כלל.

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

AllSigned

סקריפטים יכולים רוץ רק בתנאי שהם חתומים דיגיטלית ע"י מפיץ מורשה (trusted publisher), מתקבלת התראה לפני הרצת סקריפט.

RemoteSigned

סקריפטים שרצים באופן לוקאלי מהמחשב, ונוצרו בו, יכולים לרוץ ללא חתימה דיגיטלית. חתימה דיגילטית נדרשת רק לסקריפטים שהורדו מהאינטרנט למחשב.

לא מוצגת התראה לפני הרצת סקריפט מ trusted publisher.

Unrestricted

מריץ את כל סוגי הסקריפטים ללא דרישה לחתימה דיגיטלית, אולם מתריע לפני הרצת סקריפט שהגיע מהאינטרנט.

במידה והורד סקריפט מהאינטרנט ורוצים לשחרר אותו, כך שניתן יהיה להרצה ללא המגבלות של סקריפט שהגיע מהאינטרנט, יש ללכת למאפייני הקובץ וללחוץ על Unblock.

הרצת סקריפט

על-מנת להריץ סקריפט, יש לעבוד לפי סינטקס הבא:

. *<script path>*

במידה והסקריפט באותה תיקייה שאלייה מכוון PowerShell:

./<script path>

סימן ה . (dot) משייך את ה scope של הסקריפט ל scope הנוכחי. במקום הכיתוב המסומן בצהוב, יש לשים את נתיב הקובץ. במידה והסקריפט הוא זה שמבצע את הפעולה, הוא יבצע את הפעולות המוגדרות ויסיים את פעולתו.
אם הסקריפט מכיל פונקציות, ניתן לאחר מכן לקרוא לפונקציות בנפרד.

במקרה של הרצה מ run של Windows, יש להשתמש ב syntax הבא:

powershell.exe –noexit *<script path>*

הפרמטר noexit מיועדת שחלון ה console לא ייסגר מיד עם סיום הרצת הסקריפט.

הגדרת פונקציה

```
Function FunctionName
{
    #ניתן לשלב גם הגדרה של פרמטרים
    #Code
    return value(optional)
}
```

הגדרת פרמטרים
ניתן להגדיר פרמטרים לסקריפט או לפונקציה. ההגדרה מתבצעת בצורה הזאת:

```
param([string]$dllPath,[int]$num)
```

את ההגדרה הזאת יש לשים בתחילת הפונקציה או הסקריפט.
במקרה הזה, הפרמטרים שמתקבלים הם משתנה בשם dllPath מסוג string ומשתנה בשם num מסוג int.

את ההעברה יש לבצע לפי השם של המשתנים במקרה הזה (הגדרה לפי שם), ולכן, קריאה לפונקציה הזאת עם העברת הפרמטרים תתבצע כך:

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

FunctionName –dllPath *string* –num *int*

ניתן גם להגדיר ערכי ברירת מחדל:

```
Param(
[string]$str,
[string]importantDate=$throw( "You must enter Date")
)
```

במקרה זה, ערך ברירת המחדל של importantData היא פעולה של זריקת exception שתבוצע במידה ולא יועבר ערך אחר.

ניתן גם להגדיר את הפרמטרים שיהיו מבוססי position, למשל:

```
Function minus([int]$num1,[int]$num2){
return $num1 - $num2
}
```

במקרה זה, הקריאה תיראה כך:

```
>minus 10  5
>5
```

כברירת מחדל, העברת ערכים היא by value. אם רוצים להעביר ערך by reference, יש לשים לפניו "[ref]". למשל, כך תיראה הגדרת הפרמטרים:

```
param([ref]$str,[string]$str2)
```

כאן הפרמטר str יועבר by reference (אסור לציין type בנוסף ל [ref]) ואילו הפרמטר str2 מסוג string יועבר by value.

כמו כן, ניתן לעבוד עם האובייקט $Args שמכיל את הפרמטרים שהועברו לפי ה position שלהם. למשל, $Args[0] מכיל את הפרמטר הראשון שהועבר לאחר שם הפונקציה.

לולאת While

```
while(condition)
{
        #פה ניתן לשים את הקוד
        #לא לשכוח קידום בסוף...
}
$i=0
While($i –lt 100){
$i
$i++
}
```

הקוד הזה ידפיס את המספרים מ-0 עד 99.

**הסימנים קטן מ- ו גדול מ- נקראים עכשיו lt– ו gt– בהתאמה*

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

לולאת for

```
For(הגדרת קידום;הגדרת תנאי;הגדרת משתנה מעקב){
#Body
}
```

לדוגמא, קוד שידפיס את המספרים מ-0 עד 99:

```
for($i=0;$i -lt 100;$i++){
$i
```

לולאת foreach

לולאת foreach מיועדת למעבר על פריטים במערך או ב collection:

```
foreach($i in get-process)
{$i.Name; $i.Company;}
```

הקוד הזה ידפיס עבור כל Process שרץ כעת את שמו, ואת שם החברה שרשומה כיצרנית. מאחר ש -get process מחזירה מערך, $i הוא ה iterator שמסמן את הפריט הבודד עליו פועלים כעת.

Switch

```
$a = "Shahar"
switch ($a)
}
        "Yosi"
        }
                "The Name is Yosi"
                break
        {
        "Shahar"
        }
                "The name is Shahar"
                break
        {
        default
        }
                "Nice Name"
        {
}
```

משפט ה break הוא אופציונאלי. בין switches שניתן להעביר ל switch יש את:

-regex : מציין שהמקרים שמולם נבדק המשתנה הם למעשה מכילים ביטוי רגולרי כלשהו.

-wildcard : מציין שהמקרים מולם נבדק המשתנה מכילים למעשה מחרוזת של wildcard

-casesensitive : מציין שיש רגישות לאותיות גדולות

-exact (ברירת מחדל): התאמה מדוייקת נדרשת.

למשל:

```
$a="shahar"
switch -wildcard ($a)
{
        s*
        {
```

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

```
                "The Name begin with S"
        }
        Sh*
        {
                "The name begin with SH"

        }
        default
        {
                "Nice Name"
        }
}
```

בדוגמא הזאת נבדקים שני ביטויי wildcard והתאמתם למשתנה ההתחלתי.

קבלת ערך מהמשתמש

קבלת ערך מהמשתמש מבוצעת באמצעות read-host:

```
$userInput = read-host "Please Enter something"
```

ריבוי פקודות בשורה אחת

בשביל להכניס מספר פקודות בשורה אחת, יש להכניס אותן ולהפריד ב ; (נקודה-פסיק):

```
$a=1,2,3,4,5;$b=$a[3];write-host $b
```

קריאת קובץ טקסט

כדי לקרוא קובץ טקסט, יש להשתמש ב get-content:

```
$file = c:\file.txt
```

כעת, המערך file, מכיל בכל תא למעשה שורה אחת מהקובץ file.txt כאשר, במידה ורוצים לקבל את השורה האחרונה, יש לכתוב זאת כך:

```
$file[-1]
```

הוספת Reference של NET. לסקריפט

על-מנת להוסיף רפרנס של NET. לסקריפט של Windows PowerShell יש להשתמש בקוד הבא:

```
[System.Reflection.Assembly]::LoadWithPartialName("System.windows.forms")
```

בקוד הזה, למשל, אנחנו מוסיפים reference באמצעות המתודה הסטטית System.Reflection.Assembly.LoadWithPartialName שמאפשרת לנו להוסיף reference כאשר ה assembly עצמו רשום, באמצעות שמו בלבד.

במידה ואין אנו יודעים את השם, אלא רק את הנתיב לקובץ,

```
[System.Reflection.Assembly]::LoadFile("C:\windows\system32\inetsrv\microsoft.web.administration.dll")
```

במקרה זה ייטען ה DLL שנמצא בנתיב שצויין. שימו לב, שבמקרה כזה, כדאי לוודא שאכן הDLL נמצא במיקום הזה. לשם כך ניתן להשתמש ב cmdlet בשם test-path שמקבלת פרמטר בשם path– מסוג string שמכיל את הנתיב. הפרמטר הזה מיוחס גם ל position 0:

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

```
test-path C:\windows\system32\inetsrv\microsoft.web.administration.dll
```
יחזיר True או False בהתאם לקיום הקובץ.

בחירת properties מסויימים מאובייקט

על מנת לבחור properties מסויימים, ניתן להשתמש ב select-object. למשל:

```
$a = Get-Process | Select-Object Name, CPU
```

בדוגמא הזאת, האובייקט שחוזר ל a, מכיל רק את ה properties המסויימים הללו.

דרך אגב, במקרה זה, האובייקט שחוזר אינו System.Diagnostics.Process[] אלא:

```
System.Management.Automation.PSCustomObject
```

עבודה מול WMI

על מנת לעבוד מול WMI, יש להשתמש ב Get-WMIObject.

```
Get-WMIObject Win32_BIOS
```

כשלא מציינים namespace, אוטומטית נשלף המידע מ cimv2.
במידה שידוע שהמידע נמצא במקום אחר, יש להעביר את הפרמטר namespace:

```
Get-WMIObject SystemRestore -namespace root/default
```

Begin…Process….End

כאשר אנו כותבים cmdlet כחלק מ SnapIn, יש לנו אפשרות לדרוס אחת או יותר מתוך 3 מתודות – BeginProcessing, ProcessRecord ו- EndProcessing

כאשר כותבים script ניתן לחלק את הקוד באמת לחלק של begin, process ו-end ובדומה ל cmdlet, הקוד ב begin וב end יתבצע פעם אחת (במקרה של פעולה תקינה), ואילו הקוד ב process יתבצע

```
Begin
{
  #Code
}
Process
{
  #Code
}
End
{
  #Code
}
```

במידה ורוצים להתייחס בחלק ה process לאובייקט הנוכחי שעומד ב pipeline, יש להשתמש ב $_

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

משתנים שמוגדרים תמיד ב PowerShell

ב Windows PowerShell יש מספר משתנים קבועים, שמוגדרים תמיד ומכילים ערכים שנחוצים לעבודה.
הטבלה הבאה מכילה את המשתנים הקיימים תמיד בהם יש הכי הרבה שימוש.

שם	ערך
$^	מכיל את הביטוי הראשון שהוכנס בשורת הקלט האחרונה
$$	מכיל את הביטוי האחרון שהוכנס בשורת הקלט האחרונה
$_	האובייקט הנוכחי ב pipeline – משמש בעיקר כשעוברים על מספר רב של אובייקטים, למשל: בסקריפטים, פונקציות, filters (גם כחלק מ WhereObject cmdlet)
$?	מכיל משתנה בוליאני שמציין האם ביצוע השורה האחרונה הצליח.
$Args	מערך המכיל את כל הפרמטרים שהועברו לפי position לפונקציה או סקריפט
$Error	Collection שמכיל את השגיאות האחרונות שאירעו במסגרת ה session הנוכחי של PowerShell.
$true	מכיל את הייצוג לערך בוליאני אמת
$false	מכיל את הייצוג לערך הבוליאני שקר
$null	מכיל את הייצוג ל System.Null (לריק ולבלתי מוקצה)
$this	מציין את האובייקט הנוכחי ומשמש בעיקר בקבצי type.ps1xml
$foreach	מפנה ל enumerator בלולאת foreach.
$home	מפנה לתיקיית המשתמש הראשית - %HomePath%
$match	Hash table שמכיל את הפריטים שנמצאו מתאימים לאחר שימוש באופרטור -match
$MyInvocation	מכיל מידע לגבי הפקודה או הסקריפט הרצים כעת.
$PSHome	התיקייה אליה הותקן PowerShell
$Host	אובייקט המכיל מידע על המארח הנוכחי של ה powershell – יכול להיות ה host הדיפולטי אם רצים משורת הפקודה המקורית, או לחלופין host אחר.
$LastExitCode	מכיל מספר 0 או 1 המציין את רמת ההצלחה של ביצוע שורת הפקודה האחרונה
$ShellId	מכיל את המזהה של ה shell הנוכחי
$StackTrace	מכיל את ה Stack Trace של השגיאה האחרונה שאירעה.
$Profile	מכיל את הנתיב לקובץ הפרופיל של המשתמש. קובץ זה הוא קובץ ps1, קובץ סקריפט, שיכול להכיל פונקציות והגדרות הרלוונטיות למשתמש. קובץ זה מורץ מיד עם הפעלת ה PowerShell, ולכן ניתן לשלב בו פונקציות שמשמשות את המשתמש, filters והוספת ה SnapIns בהם נעשה שימוש.
$Input	משמש בסקריפים ופונקציות שבאמצע ה Pipeline לקבלת הקלט הבסיסי
$DebugPreference	מציין מה לבצע במקרה שיש שימוש ב write-debug cmdlet
$MaximumAliasCount	מציין את מקסימום ה aliases (כינויים, למשל dir הוא alias ל -get childitem) שניתן להשתמש בהם ב session.
$MaximumDriveCount	מציין את כמות הכוננים שניתן ליצור באמצעות new-psdrive
$MaximumFunctionCount	מציין את כמות הפונקציות המקסימלית ב session אחד.
$MaximumVariableCount	מציין את כמות המשתנים המקסימלית ב session אחד.
$VerbosePreference	מציין מה הפעולה שתתבצע בעת כתיבת verbose
$WarningPreference	מציין מה הפעולה שתתבצע בעת כתיבת warning
$ErrorActionPreference	מציין מה הפעולה שתתבצע בעת כתיבת error

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

מוסכמות פיתוח

בחלק זה יובאו מסכמות בנושא פיתוח cmdlets. המוסכמות הללו מתחלקות ל3 קטגוריות: הכרחי – כללים שאי יישומם ימנע שימוש ב cmdlet או יקשה מאד על שימוש כזה, מומלץ – כללים שכדאי מאד לבצע, ורצוי – כללים שצריך להפעיל שיקול דעת לגבי ביצועם לפי המקרה.

הכרחי

רש מ Cmdlet או PSCmdlet

חובה לרשת מ System.Management.Automation.Cmdlet או מ PSCmdlet.

הגדר CmdletAttribute

חובה לציין בראש ה class שמהווה את ה cmdlet את CmdletAttribute שם אתה מגדיר בין השאר את שם ה Cmdlet.

עליך להקפיד:

- הפועל ושם-העצם באמת מתארים בצורה נכונה וברורה את ה Cmdlet
- במקרה שה cmdlet מקבל יותר מפרמטר אחד, ציינת מהו פרמטר ברירת המחדל.
- אתה תומך ב ShouldProcess במידה ואתה מבצע פעולה שעלולה לדרוש את את אישור המשתמש ומבצעת שינויים במערכת.

השתמש רק בפעלים מוכרים

על אף שאת ה verb ניתן להעביר בתור string יש להשתמש בפועלים מהרשימה המובנת – למשל, ב VerbsCommon או VerbsCommunication

ביצעת override לאחת ממתודות הפעולה

ביצעת override ללפחות אחת ממתודות הפעולה: BeginProcessing, ProcessRecord, EndProcessing

יצירת SnapIn לצורך התקנת ה cmdlet

חובה ליצור SnapIn שיורש מ PSSnapIn או מ CustomPSSnapIn ועושה override לנתונים האבסטרקטיים שבהם. חובה להקפיד ששם ה SnapIn אינו מכיל רווחים ואינו מכיל את התווים הבאים: # . , ( ) { } [ ] & - \/ $; : " ' < > | ? @ ` *

תמוך ב ShouldProcess

לפני ביצוע שינויים במערכת, או לפני ביצוע פעולות המשנות את המערכת, קרא למתודה ShouldProcess ופעול בהתאם לערך הבוליאני שהיא מחזירה. זה יאפשר לסביבת הריצה של PowerShell להשתמש ב confirm– וב – whatif עבור ה Cmdlet שלך.

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

טפל בשגיאות לפי הכללים

השתדל לטפל בשגיאות לפי הכללים הבאים:

- כאשר אירעה שגיאה שמונעת את המשך הטיפול בשאר המידע שמועבר מה pipeline, על ה cmdlet לקרוא למתודה ThrowTerminatingError ולהעביר אליה אובייקט ErrorRecord עם מידע השגיאה הרלוונטי.
- אם אירעה שגיאה שלא דורשת את הפסקת פעולת ה cmdlets עבור שאר האובייקטים הממתינים ב pipeline (למשל: שגיאה הנוגעת לקובץ אחד מתוך רשימה ארוכה של קבצים, ולא משפיעה על המשך הפעולה) יש לקרוא למתודה WriteError. זאת על אף, שבמידה ולא נקראה המתודה, Windows PowerShell יתפוס את ה exception
- **לתשומת לבך:** עם ה cmdlet קרא לthread אחר וביצע שם פעולה שגרמה לזריקת Exception שלא נתפס ע"י ה cmdlet, הוא לא ייתפס גם ע"י PowerShell והפעולה תופסק לחלוטין.
באותה מידה לגבי קוד שנמצא במסגרת destructor או במסגרת מתודת Dispose

מומלץ

מומלץ להקפיד על כללי מתן השם לשם עצם

- רצוי ששם העצם יהיה ביחיד.
- רצוי שיהיה ברור ומדוייק

פרמטרים

בעת הגדרת פרמטרים, עליך להגדיר Property ציבורי (public) ולציין לפניו ParameterAttribute.

מומלץ לתת שמות הגיוניים, לפי שיטת פסקל (כל מילה מתחילה באותיות גדולות).

רצוי לצרף את הפרמטרים ל ParameterSet כדי לאפשר להגדיר ParameterSe t דיפולטי.

פרמטרים שערכם נבחר ממספר אפשרויות

ישנן שני דרכים עיקריות ליצור פרמטר שערכו נבחר מתוך רשימת ערכים אפשריים:

1. שימוש ב enum שמכיל את הערכים שיכולים להתקבל, וקבלת בחירה מסוג ה enum
2. שימוש ב ValidateSetAttribute לפני הגדרת הפרמטר.

פרמטרים שמקבל ערך בוליאני (אמת\שקר)

רצוי להגדיר את ה type ל SwitchParameter ולא ערך בוליאני. ברגע שיופיע ה switch זה ייחשב לאמת, אחרת לשקר. נניח, למשל, פרמטר מסוג SwitchParameter בשם CaseSensetive ב cmdlet ששמו get-files:

```
Get-files –CaseSensetive
```

בשורה הזאת, לדוגמא, הערך הוא true, כי יש איזכור של ה switch

אל תשתמש ב System.Console

שימוש ב System.Console "כובל" את ה cmdlets שלך לפעול רק דרך שורת הפקודה, מקשה על מפתחים שרוצים להתממשק אליהם ולקבל אובייקטים ומונע שימוש דרך hosts שאינם מבוססי command line.

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

אל תשתמש בפונקציונאליות שייתכן שאינה נתמכת תמיד

ה Cmdlet מגיע עם אפשרויות להתממשק לחלקים שונים של ה host. למשל, השורה הבאה, תשנה את צבע הרקע של הפלט:

```
this.CommandRuntime.Host.UI.RawUI.BackgroundColor = ConsoleColor.Cyan;
```

אולם, במידה וה host לא ממש פונקציונאליות זאת, ייזרק Exception. לעיתים, יש פונקציונאליות שה host בכלל לא יכול לממשל, נניח, הקוד הבא:

```
this.CommandRuntime.Host.UI.ReadLine();
```

אם ה host הוא, נניח, WPF Application, הוא עלול להתקשות לממש את זה, מה שגם יגרום ל Exception. לכן, מומלץ להימנע משימוש בפונקציונאליות שייתכן שחלק מהמארחים של ה cmdlet לא יוכלו לבצע ולא מימשו.

תמוך במנגנון ה Path של Powershell

בשל מנגנון ה Providers המשוכלל, והאפשרות ליצור "כוננים" שונים, רצוי לתמוך בשיטת ה Path של PowerShell. המתודות הבאות יכולות לשמש לכך (שים לב, שהמתודות שנמצאות ב PSCmdlet רלוונטיות רק לקמרה שירשת מ PSCmdlet:

```
System.Management.Automation.PSCmdlet.GetResolvedProviderPathFromPSPath(System.String,System.Management.Automation.ProviderInfo)
System.Management.Automation.PSCmdlet.GetUnresolvedProviderPathFromPSPath(System.String)
System.Management.Automation.PathIntrinsics.GetResolvedProviderPathFromPSPath(System.String,System.Management.Automation.ProviderInfo)
System.Management.Automation.PathIntrinsics.GetUnresolvedProviderPathFromPSPath(System.String)
```

תמוך ב Wildcard

בפרמטרים, השתדל לתמוך ב wildcard שמאפשר למשתמש לציין תבנית של הערך אותו הם מעבירים, התמיכה קלה עד מאד – באמצעות System.Management.Automation.WildcardPattern ושימוש במתודה WildcardPattern.IsMatch לבדיקת התאמה. שים לב, שה constructor של WildcardPattern מקבל את המחרוזת שמכילה את התנאי (ה pattern) ויכול לקבל גם WildcardOptions. מומלץ להגדיר WildcardOptions.IgnoreCase כדי למנוע את הרגישות לאותיות רישיות (Case Insensitive). במידה ומגדירים WildcardOptions.Compiled – ה wildcard pattern יקומפל לתוך ה assembly, מה שישפר את זמן ההרצה, אולם עלול להאריך במעט את זמן הטעינה.

קבל מערכים בתור פרמטרים

השתדל לקבל מערכים בתור פרמטרים, כדי להקל על המשתמש ולאפשר לו לבצע מספר פעולות ב cmdlet אחד. למשל, ב cmdlet ששמו stop-websites שהוצג בהרצאה, ניתן להעביר בבת אחת מספר שמות של websites כדי לעצור.

תמוך ב passthru–

ה passthru switch– מאפשר לאחר ביצוע פעולה על ערך מסויים שהועבר, להעביר אותו ל pipeline להמשך טיפול. למשל:

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

Stop-process winword –passthru

במקרה זה, ה- process יופסק אבל יועבר ל pipeline למקרה שה cmdlet הבא מעוניין לקבל אותו ולעבוד עליו.

ממש את IComparable

מימוש זה יקל על Windows PowerShell לנהל את האובייקטים שאתה מחזיר ב pipeline. מדובר על אובייקטים שמחזירים באמצעות WriteObject

הגדר תצוגה ברירת מחדל

במידה ואתה משתמש באובייקט שלא מוגדרת עבורו תצוגה ברירת-מחדל, או שהתצוגה אינה מתאימה לך, באפשרותך ליצור קובץ formats.ps1xml ולרשום אותו, דרך ה SnapIn או דרך update-formatdata כדי להגדיר את מאפייני התצוגה.

תמוך בקלט מה Pipeline

בפרמטרים שאתה מגדיר, ניתן להגדיר ב ParameterAttribute ערכים כמו ValueFromPipeline או ValueFromPipelineByPropertyName – אפשר ערכים אלה. במידה ולא תאפשר, לא ניתן יהיה להשתמש ב cmdlet שלך בצורה נוחה כחלק מ pipeline.

ממש אתProcessRecord

ה Cmdlet שלך חייב לממש אחת משלושת מתודות הפעולה – אולם, רצוי שתמיד תממש את ProcessRecors כדי לתמוך בפעולה מרובת פעמים, בהתאם לערכים שמגיעים דר ה pipeline.

עבוד בצורה נכונה עם WriteObject

הקפד תמיד כשכותבים Collection, להעביר בנוסף ערך בוליאני true למתודה, לפי ה overload – WriteObject(object,bool).

בנוסף, הקפד להשתמש ב WriteObject רק מה thread הראשי של ה cmdlet אחרת ייזרק InvalidOperationException.

תהיה CaseInsensitive

Windows PowerShell אינו רגיל לגודל האות. השתדל שתמיד, ב cmdlets שאתה כותב, גם לא תהיה רגישות כזאת. בכל אופן, במידה ויש רגישות לאות גדולה\קטנה, הערך שמועבר יהיה כפי שהוכנס ע"י המשתמש או הועבר ב pipeline – PowerShell לא מבצע שום המרות.

עבוד בצורה נכונה עם ShouldProcess ועם ShouldContinue

השתמש ב ShouldProcess כדי לתמוך בפרמטר whatif–

בצורה כזאת, כאשר המשתמש יכניס את הפרמטר whatif– באופן אוטומטי הוא יקבל, באמצעות הקריאות בקוד ל ShouldProcess; מידע על הפעולות שיתבצעו. אם הוא יעביר את הפרמטר confirm– הוא יידרש גם לאשר בפועל.

שימוש ב ShouldContinue יציג תמיד הודעה, ולכן יש להשתמש בו רק במקרים מיוחדים, כדי לא להטריח את המשתמש. בכל אופן, צריך תמיד לקבל פרמטר מסוג SwitchParameter בשם Force שיאפשר להחליט בצורה גורפת על ביצוע הפעולה.

קוד נכון שמשתמש ב ShouldContinute ייראה כך:

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

```
if(ShouldProcess(…))
{
        if(Force || ShouldContinue(…))
        {
                //Code
        }

}
```

השתמש ב WriteWarning, WriteVerbose ו- WriteDebug

מתודות אלו, מאפשרות לך אינטראקציה נוספת ועירוב נוסף של המשתמש.
קריאה ל **WriteWarning** צריכה להתבצע במקרה שעומדת להתבצע פעולה שעלולה לגרום לתוצאות לא צפויות.
שימוש ב **WriteVerbose** יציג את המידע רק כאשר הועבר המרפטר verbose– ל cmdlet.
יש לשים שם מידע נוסף, שעלול לעניין את המשתמש שביקש להיות יותר מעורב בתהליך.
WriteDebug משמש כשנדרשת רמה גבוהה אף יותר של פירוט, נניח ע"י מפתח אחר או אדם שבוחן את המוצר ומנסה לנתח בעייה שאירעה במהלך השימוש. אין צורך להכניס מידע גם ב WriteVerbose וגם ב- WriteDebug מאחר שקריאה ל cmdlet עם הפרמטר debug– תביא גם מידע במסגרת WriteVerbose.

בפעולות ארוכות השתמש ב WriteProgress

השתמש ב WriteProgress על מנת לציין תהליך והתקדמות בפעולות העלולות לקחת זמן ארוך.

רצוי

הגדר InputObject

הגדר פרמטר בשם InputObject מה type של האובייקט שאותו אתה רוצה להחזיר.

הגדר Namespace

הגדר את כל ה cmdlets שלך ב אותו namespace, רצוי כזה שיהיה בסופו Commands. למשל, ה cmdlets של מיקרוסופט, נמצאים ב Microsoft.PowerShell.Commands

השתמש במנגנון הזהות של PowerShell

אם אתה צריך לקבל שם משתמש וסיסמא, הגדר פרמטר מסוג PSCredential והגדר CredentialAttribute שיאפשר הצגת תיבת שם משתמש וסיסמא למשתמש.

דרוס את StopProcessing

דרוס את המתודה StopProcessing ושים שם את הקוד שיתרחש במקרה של הפסקה לא צפוייה של פעולת ה cmdlet.

ממש את IDisposable

מימוש של IDisposable יאפשר ל PowerShell לטפל בצורה יעילה יותר בפינוי משאבים של ה cmdlet שלך.

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net

במידה ואין ערכי Pipeline השתמש ב BeginProcessing

בד"כ כשאמורים להתקבל מספר ערכים ב pipeline, רצוי לשים את הלוגיקה המרכזית ב ProcessRecord שיכול לפעול מספר פעמים, לפי כמות הערכים המועברים.

במידה ולא עומדים להיות מועברים ערכים ב Pipeline, עדיף, מבחינת יעילות ומהירות הפעולה, לדרוס רק את Cmdlet.BeginProcessing ולשים שם את כל הלוגיקה.

התקנת PowerShell

Installutil.exe מגיע כחלק מ NET Framework 2.0 SDK. ומאפשר התקנה של רכיבים שונים, ובהם PowerShell Snapins. כאשר אתם ב windows powershell ומנסים להתקים את ה cmdlet, מומלץ לכם לקבוע איזשהו alias שיפנה לנתיב של הקובץ, למשל:

```
Set-alias installutil $env:windir\Microsoft.NET\Framework\v2.0.50727\installutil
```

בצורה זאת, אתם מגדירים alias בשם installutil שמפנה לנתיב של הקובץ עצמו. כעת, יש להשתמש ב:

installutil *dllPath*

כאשר dllPath הוא הנתיב לקובץ הDLL.

על-מנת להסיר, יש להשתמש בsyntax הבא:

installutil /u *dllPath*

לאחר ההתקנה יש להוסיף את ה snapin באמצעות ה cmdlet ששמו add-pssnapin.

מידע נוסף

הבלוג של שחר גבירץ – http://blogs.microsoft.co.il/blogs/shahar
הבלוג של צוות Windows PowerShell – http://blogs.msdn.com/powershell
מרכז הסקריפטים של PowerShell באתר TechNet -
http://www.microsoft.com/technet/scriptcenter/hubs/msh.mspx

©כל הזכויות שמורות ל **שחר גבירץ**.

http://blogs.microsoft.co.il/blogs/shahar
Shahar@IShahar.net