Windows PowerShell 3.0 Examples

Created by http://powershellmagazine.com

PowerShellMagazine

How to schedule a job

Job scheduling allows you to schedule execution of a PowerShell background job for a later time. First thing you do is create a job trigger. This defines when the job will execute. Then you use the Register-ScheduledJob cmdlet to actually register the job on the system. In the following example, every day at 3am we back up our important files:

```
$trigger = New-JobTrigger -Daily -At 3am
Register-ScheduledJob -Name DailyBackup -Trigger $trigger -ScriptBlock {Copy-Item
c:\ImportantFiles d:\Backup$((Get-Date).ToFileTime()) -Recurse -Force -PassThru}
```

Once the trigger has fired and the job has run, you can work with it the same way you do with regular background jobs:

```
Import-Module PSScheduledJob
Get-Job -Name DailyBackup | Receive-Job
```

You can start a scheduled job manually, rather than waiting for the trigger to fire:

```
Start-Job -DefinitionName DailyBackup
```

How to generate complex password

There is no built-in cmdlet to generate a password, but we can leverage a vast number of .NET Framework classes. System.Web.Security.Membership class has a static method GeneratePassword(). First, we need to load an assembly containing this class, and then we use GeneratePassword() to define password length and a number of non-alphanumeric characters.

```
$Assembly = Add-Type -AssemblyName System.Web
[System.Web.Security.Membership]::GeneratePassword(8,3)
```

How we know all that? With a little help of Get-Member cmdlet (and simplified where syntax):

```
[System.Web.Security.Membership]  | Get-Member -MemberType method -Static|
where name -match password
```

How to convert WMI dates

By default, Get-WmiObject returns information about date properties in a format that's not easy to read. ConvertToDateTime() method to the rescue! In the following example you will see a lot of aliases-cn is an alias for ComputerName parameter, foreach is an alias for ForEach-Object cmdlet, and $PSItem is a new automatic variable and an alias for $_ automatic variable.

```
Get-WmiObject Win32_OperatingSystem -cn localhost |
 foreach {$PSItem.ConvertToDateTime($PSItem.LastBootUpTime)}
```

Get-CimInstance, a new CIM cmdlet similar to Get-WmiObject returns date properties in a readable fomat:

```
Get-CimInstance -ClassName Win32_OperatingSystem | fl last*
```

How to install PowerShell Web Access (PSWA)

PowerShell Web Access is a new feature on Windows Server 2012. It acts as a gateway, and from there you can run PowerShell commands on any machines in your environment. The webpage is entirely Javascript-based, so you can access the console on almost any device with a web browser. You can install and configure PSWA for testing purposes in three easy steps:

```
Install-WindowsFeature -Name WindowsPowerShellWebAccess -Verbose
Install-PswaApplication -UseTestCertificate
Add-PswaAutorizationRule -UserName * -ComputerName * -ConfigurationName *

# start your default web browser and log in
Start-Process https:/dc.contoso.com/pswa
```

How to create a share in Windows Server 2012

Quite easy, with a one-liner. Windows Server 2012 provides Windows PowerShell cmdlets and WMI objects to manage SMB file servers and SMB file shares. The SMB cmdlets are packaged into two modules called SmbShare and SmbWitness. These modules are automatically loaded (thanks to new module auto-loading feature) whenever you refer to any of the cmdlets included. No upfront configuration is required. Note: Check the output of Get-Module command before and after you run the following one-liner to see that SmbShare module is loaded behind the scenes.)

```
New-SmbShare –Name MyShare –Path C:\Test –Description 'Test Shared Folder' –
FullAccess Administrator –ChangeAccess DouglasAdams -ReadAccess Everyone
```

Windows PowerShell 3.0 Examples

Created by http://powershellmagazine.com

PowerShellMagazine

How to parse web page elements

The Invoke-WebRequest parses the response, exposing collections of forms, links, images, and other significant HTML elements. The following example returns all the URLs from a Web Page:

```
$iwr = Invoke-WebRequest http://blogs.msdn.com/b/powershell
$iwr.Links
```

The next example returns an RSS feed from PowerShell team blog:

```
$irm = Invoke-RestMethod blogs.msdn.com/b/powershell/rss.aspx
$irm | Select-Object PubDate, Title
```

How to fetch data exposed by OData services

New Invoke-RestMethod cmdlet sends an HTTP(S) request to a REST-compliant web service. You can use it to consume data exposed by, for example, OData service for TechEd North America 2012 content. Put information about TechEd NA 2012 sessions into $sessions variable. Let's iterate through that collection of XML elements and create custom PowerShell objects using [PSCustomObject] (a new way to do it). Pipe the output to Out-GridView command specifying PassThru parameter to send selected items from the interactive window down the pipeline as input to Export-Csv command. (While you are in Out-GridView window, try to filter only PowerShell Hands-on Labs, for example.). If you have Excel installed the last command will open CSV file in Excel.

```
$sessions = Invoke-RestMethod http://odata.msteched.com/tena2012/sessions.svc/
Sessions

$sessions | ForEach-Object {
   $properties = $_.content.properties
   [PSCustomObject]@{
      Code      = $properties.Code
      StartTime = $properties.StartTime.'#text'
      EndTime   = $properties.EndTime.'#text'
      Title     = $properties.Title
      Abstract  = $properties.Abstract
   }
} | Out-GridView -PassThru | Export-Csv $env:TEMP\sessions.csv -
NoTypeInformation

Invoke-Item $env:TEMP\sessions.csv
```

How to set custom default values for the parameters

If you, for example, need to execute one of the -PSSession cmdlets using alternate credentials you must specify the credentials for the Credential parameter every time you call the command. In v3, we can supply alternate credentials by using new $PSDefaultParameterValues preference variable. For more information, see about_Parameters_Default_Values.

```
$PSDefaultParameterValues = @{'*-PSSession:Credential'=Get-Credential}
```

How to easily add multiple note properties to an object

With the new enhancements to the Add-Member cmdlet we can now use a hash table to add multiple note properties to an object:

```
New-Object -TypeName PSObject | Add-Member @{One=1; Two=2; Three=3} -
PassThru
```

How to leverage "implicit foreach" feature

New "implicit foreach" feature allows you to access properties and methods on objects inside of a collection.

In v2, the following would give no result:

```
(Get-Process).Name
```

In v3, PowerShell returns all process names. We can also access "properties of properties"

```
(Get-Process).StartTime.DayOfWeek
```

It works with methods too (remove all instances of Notepad):

```
(Get-Process notepad).Kill()
```

How to access local variables in a remote session

In v2, if you had some local variable that you wanted to use when executing a script block remotely, you had to do something like:

```
$localVar = 42
Invoke-Command -ComputerName Server 1 { param($localVar) echo $localVar } -
ArgumentList $localVar
```

In v3, you can use Using scope (prefix variable name with $using:):

```
Invoke-Command -ComputerName Server1 { echo $using:localVar }
```