Creating Data VHD using PowerShell

Sivaprasad Padisetty (November 2013)

*If you haven’t used PowerShell automation for Hyper-V, it is time to try it. This is a short blog that solved couple of tricky and annoying issues that blocks automation. Creating a data VHD (or VHDX) using PowerShell is tricky, requires connecting a few cmdlets together. An annoying popup “You need to format the disk in drive D: before you can use it.” breaks automation. Proposed solution solves this problem.*

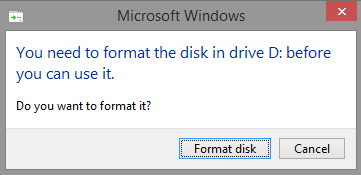
*In Windows 8.1/Windows Server 2012 R2, Mounting a VHD, sometimes the drive letter does not show up. The code fragment below has a workaround to solve this as well.*

# 

# Create the VHD file

VHD file can be used as a generic container to store file collection. The file collection can be hierarchical (i.e.) like a disk drive. Hence it can be used as a packaging mechanism, easy to transfer single file. This also comes in handy with automation. A typical automation scenario is: Create a VHD file, copy the file content to this VHD file, and make this VHD available to the VM as an additional disk. This technique is used in [multi](http://multi.codeplex.com).

New-VHD cmdlet creates a VHD file, but the created file is not formatted (hence no file system). Before the disk is formatted, it has to be initialized and a partition should be created. After I figured how to format the disk, I started to hit the following message box randomly.



I tried different things with no luck. Finally, I reached out to Senthil Rajaram, Senior Program Manager at Microsoft. He gave me the brilliant code fragment that solved this annoying popup.

# Creates a data VHD file (non OS VHD file)

# VHDPath – Full path to a VHD file

# Size – size of the VHD file to create. This creates a dynamic VHD,

# (i.e) the VHD file is not expanded to full size

function CreateVHD ($VHDPath, $Size)

{

$drive = (New-VHD -path $vhdpath -SizeBytes $size -Dynamic | `

Mount-VHD -Passthru | `

get-disk -number {$\_.DiskNumber} | `

Initialize-Disk -PartitionStyle MBR -PassThru | `

New-Partition -UseMaximumSize -AssignDriveLetter:$False -MbrType IFS | `

Format-Volume -Confirm:$false -FileSystem NTFS -force | `

get-partition | `

Add-PartitionAccessPath -AssignDriveLetter -PassThru | `

get-volume).DriveLetter

Dismount-VHD $VHDPath

}

Note: In the above code sample, the expression to create VHD and format is a single pipeline, but spread over multiple lines for readability. If a line ends with the character “`”, it tells PowerShell that the following line is a continuation of the current line.

# MountVHD and DismountVHD

To access and manipulate a VHD file, it has to be mounted. Use the Mount-VHD cmdlet to mount the VHD file. By default it assigns a drive letter to the mounted VHD. (i.e.) You can access the VHD file like a disk drive. I wrote a simple helper function “MountVHD”, this takes the path to the VHD file and returns the path where the file is mounted.

# Mount the VHD file so that it can be accessed like a drive

# $VHDPath points to VHD file (full path)

# Returns path to the mounted VHD (e.g.) D:\

function MountVHD ($VHDPath)

{

Mount-VHD $VHDPath

$drive = (Get-DiskImage -ImagePath $VHDPath | `

Get-Disk | `

Get-Partition).DriveLetter

"$($drive):\"

Get-PSDrive | Out-Null # Work around. some times the drive is not mounted

}

# Dismount an already mounted VHD file

# After dismount the drive is not accessible.

function DismountVHD ($VHDPath)

{

Dismount-VHD $VHDPath

}

# Sample Usage

Now that the functions are defined, invoke the function to create a VHD file, copy some sample files.

# Create a VHDX file c:\temp\x.vhdx

CreateVHD -VHDPath "c:\temp\x.vhdx" -Size 10GB

# Mount the VHD, copy c:\data to it and finally dismount it.

$path = MountVHD -VHDPath 'c:\temp\x.vhdx'

Copy 'c:\data' $path -Recurse

DismountVHD -VHDPath 'c:\temp\x.vhdx'

Explore & Enjoy!

/Siva