A PowerShell  
 Module for Hyper-V

Author: James O’Neill (<http://JamesOne111.worpress.com> )

This document, the PowerShell scripts with which it is supplied, and updates to them  
 can be obtained from <http://www.codeplex.com/psHyperV>

**Copyright Microsoft Corporation 2010**  
Distributed under the Microsoft Public License, MS-PL found at <http://www.codeplex.com/PsHyperV/License>

In particular, your attention is drawn to condition 3 E,  
*The software is licensed "as-is." You bear the risk of using it.  
 The contributors give no express warranties, guarantees or conditions.*

[Getting Started 6](#_Toc288174126)

[PowerShell verbs and nouns 7](#_Toc288174127)

[Standard PowerShell verbs used 7](#_Toc288174128)

[Non-standard PowerShell verbs from version 1 of the library 8](#_Toc288174129)

[PowerShell Parameters 8](#_Toc288174130)

[Summary of functions provided 9](#_Toc288174131)

[Commands in the library, grouped by function 10](#_Toc288174132)

[Virtual Machine operations 10](#_Toc288174133)

[VM Memory 10](#_Toc288174134)

[VM Processors 10](#_Toc288174135)

[VM Serial Ports 10](#_Toc288174136)

[VM Remote FX Controllers 10](#_Toc288174137)

[VM floppy disk, hard disk and DVD drives and controllers 11](#_Toc288174138)

[Virtual Hard Disk and Floppy disk images 11](#_Toc288174139)

[VM to host integration 11](#_Toc288174140)

[VM Ethernet cards 12](#_Toc288174141)

[Virtual Network Switches 12](#_Toc288174142)

[Hyper-V Servers. 12](#_Toc288174143)

[Snapshots 12](#_Toc288174144)

[Cluster Operations 12](#_Toc288174145)

[User Interface 13](#_Toc288174146)

[Miscellaneous Admin functions 13](#_Toc288174147)

[Detailed Explanation of the functions 14](#_Toc288174148)

[Start-VM 14](#_Toc288174149)

[Stop-VM 15](#_Toc288174150)

[Save-VM 16](#_Toc288174151)

[Set-VMState 17](#_Toc288174152)

[Invoke-VMShutdown 18](#_Toc288174153)

[Ping-VM 19](#_Toc288174154)

[Test-VMHeartbeat 19](#_Toc288174155)

[Get-VM 20](#_Toc288174156)

[Select-VM 21](#_Toc288174157)

[Get-VMByMACaddress 21](#_Toc288174158)

[New-VMConnectSession 22](#_Toc288174159)

[Get-VMThumbnail 23](#_Toc288174160)

[Get-VMSummary 24](#_Toc288174161)

[Get-VMBuildScript 24](#_Toc288174162)

[Get-VMSnapshot 25](#_Toc288174163)

[Get-VMSnapshotTree 26](#_Toc288174164)

[New-VMSnapshot 26](#_Toc288174165)

[Remove-VMSnapshot 27](#_Toc288174166)

[Rename-VMsnapshot 28](#_Toc288174167)

[Restore-VMSnapshot 29](#_Toc288174168)

[Select-VMSnapshot 30](#_Toc288174169)

[Update-VMSnapshot 30](#_Toc288174170)

[Remove-VMSnapshot 31](#_Toc288174171)

[Export-VM 32](#_Toc288174172)

[Import-VM 33](#_Toc288174173)

[New-VM 34](#_Toc288174174)

[Remove-VM 35](#_Toc288174175)

[Get-VMSettingData 35](#_Toc288174176)

[Set-VM 36](#_Toc288174177)

[Set-VMRASD 37](#_Toc288174178)

[New-VMRASD 37](#_Toc288174179)

[Add-VMRASD 37](#_Toc288174180)

[Remove-VMRASD 38](#_Toc288174181)

[Get-VMMemory 39](#_Toc288174182)

[Set-VMMemory 40](#_Toc288174183)

[Get-VMCPUCount 41](#_Toc288174184)

[Set-VMCPUCount 42](#_Toc288174185)

[Get-VMKVP 43](#_Toc288174186)

[Get-VMProcessor 44](#_Toc288174187)

[Get-VMSerialPort 44](#_Toc288174188)

[Set-VMSerialPort 45](#_Toc288174189)

[Get-VMIntegrationComponent 46](#_Toc288174190)

[Set-VMIntegrationComponent 46](#_Toc288174191)

[Add-VMKVP 47](#_Toc288174192)

[Remove-VMKVP 48](#_Toc288174193)

[Get-VMRemoteFXController 48](#_Toc288174194)

[Set-VMRemoteFXController 49](#_Toc288174195)

[Add-VMRemoteFXController 50](#_Toc288174196)

[Remove-VMRemoteFXController 51](#_Toc288174197)

[Get-VMNic 52](#_Toc288174198)

[Select-VMNic 53](#_Toc288174199)

[Add-VMNic 54](#_Toc288174200)

[Remove-VMNic 55](#_Toc288174201)

[Set-VMNICAddress 56](#_Toc288174202)

[Get-VMNicSwitch 57](#_Toc288174203)

[Set-VMNICSwitch 58](#_Toc288174204)

[Get-VMNicVLAN 58](#_Toc288174205)

[Set-VMNicVLAN 59](#_Toc288174206)

[Add-VMSCSIController 60](#_Toc288174207)

[Remove-VMSCSIcontroller 60](#_Toc288174208)

[Get-VMDiskController 61](#_Toc288174209)

[Get-VMDriveByController 62](#_Toc288174210)

[Add-VMDrive 62](#_Toc288174211)

[Remove-VMdrive 63](#_Toc288174212)

[Get-VMDiskByDrive 64](#_Toc288174213)

[Add-VMDisk 64](#_Toc288174214)

[Get-VMDisk 65](#_Toc288174215)

[Set-VMDisk 66](#_Toc288174216)

[Add-VMNewHardDisk 67](#_Toc288174217)

[Add-VMPassThrough 68](#_Toc288174218)

[Select-VMPhysicalDisk 68](#_Toc288174219)

[Get-VMFloppyDisk 69](#_Toc288174220)

[Add-VMFloppyDisk 70](#_Toc288174221)

[Remove-VMFloppyDisk 71](#_Toc288174222)

[New-Vfd 71](#_Toc288174223)

[New-VHD 72](#_Toc288174224)

[Get-VHDDefaultPath 72](#_Toc288174225)

[Get-VHD 73](#_Toc288174226)

[Get-VHDInfo 73](#_Toc288174227)

[Test-VHD 74](#_Toc288174228)

[Connect-VHDParent 75](#_Toc288174229)

[Compress-VHD 76](#_Toc288174230)

[Convert-VHD 77](#_Toc288174231)

[Expand-VHD 79](#_Toc288174232)

[Merge-VHD 80](#_Toc288174233)

[Mount-VHD 81](#_Toc288174234)

[Dismount-VHD 82](#_Toc288174235)

[Get-VHDMountPoint 83](#_Toc288174236)

[Wait-ForDisk 83](#_Toc288174237)

[Get-VMSwitch 83](#_Toc288174238)

[Select-VMSwitch 84](#_Toc288174239)

[Remove-VMSwitch 84](#_Toc288174240)

[New-VMPrivateSwitch 85](#_Toc288174241)

[New-VMInternalSwitch 86](#_Toc288174242)

[New-VMExternalSwitch 87](#_Toc288174243)

[Select-VMExternalEthernet 88](#_Toc288174244)

[Remove-VMSwitchNIC 88](#_Toc288174245)

[Get-VMNicPort 89](#_Toc288174246)

[New-VMSwitchPort 89](#_Toc288174247)

[Get-VMHost 90](#_Toc288174248)

[Set-VMHost 90](#_Toc288174249)

[Get-VMLiveMigrationNetwork 91](#_Toc288174250)

[Select-VMLiveMigrationNetwork 91](#_Toc288174251)

[Move-VM 92](#_Toc288174252)

[Get-VMClusterGroup 93](#_Toc288174253)

[Sync-VMClusterConfig 94](#_Toc288174254)

[Select-ClusterSharedVolume 94](#_Toc288174255)

[Test-WMIJob 95](#_Toc288174256)

[Test-WMIResult 95](#_Toc288174257)

[Select-List 96](#_Toc288174258)

[Select-Item 96](#_Toc288174259)

[Select-Enumtype 97](#_Toc288174260)

[Select-Tree 97](#_Toc288174261)

[Out-Tree 98](#_Toc288174262)

[Test-Admin 98](#_Toc288174263)

[Convert-DiskIDtoDrive 99](#_Toc288174264)

[Get-FirstAvailableDriveLetter 99](#_Toc288174265)

[A quick guide to the PowerShell code in the module 100](#_Toc288174266)

[WMI Object classes used by the functions 105](#_Toc288174267)

# Getting Started

The commands described below are implemented in the form of a PowerShell Version 2 Module which is distributed as a single .Zip file, from <http://PShyperV.codeplex.com> with this document being made available separately.

A previous version was implemented for Version 1 of PowerShell and remains available on Codeplex. The old version will work on PowerShell V2, but the new version will not work on PowerShell V1. Both versions work with the original release of Hyper-V and the R2 release; and this document reflects the changes made for R2 service pack 1.

The installation process consists of placing the unzipped PowerShell files into a suitable folder.

**BEFORE YOU UNZIP THE FILES –** downloaded files are flagged and will be treated as remote files by PowerShell. Select File Properties for the ZIP file and click the Unblock button before you go any further.

With the files expanded module can be imported into a PowerShell session using the command Import-Module <path\>Hyperv.psd1

If no path is specified, PowerShell uses the system environment variable PSModulePath and examines each folder it references to find a folder using the name specified for the module, containing a manifest file of the same name.

So, for example if you place the files into the folder c:\users\yourID\documents\windowsPowershell\modules\HyperV

you can just issue the command  
 install-module HyperV

(Install module can be shortened to IPMO).

The .Zip file includes a simple Install.cmd file which will make the following changes

* Install PowerShell on Hyper-V Server or Windows Server 2008-R2 Core installations.   
  This produces a harmless error on full installations of Server 2008 , which can be ignored.
* Create a folder under Program Files and add this folder to the PSModulePath   
  environment variable.
* Place the files in a subfolder of this folder.
* Set Registry settings to give an ideal PowerShell console window   
  and to allow PowerShell to run unsigned scripts.

You may do any or all of these things manually, and good practice says you should review a Registry file before allowing it to change your system.

**The scripts are not signed and if you wish to implement a policy of signing scripts you should verify the script for yourself and then sign it. If you do not use the installation script you should set powerShell's execution Policy to RemoteSigned with the command**Set-ExecutionPolicy RemoteSigned

The moudle includes a menu – similar to the PowerShell configurator found on Codeplex at <http://psconfig.codeplex.com/> you can launch the menu with the command Show-Hypervmenu

Alternatively when the module has been imported, you can discover its commands by entering the commands

get-command -module HyperV

or

get-command -module Hyperv| get-help | format-table name,synopsis -auto

Finally get-help *Command-Name* –full will give you the online help for a command.

# PowerShell verbs and nouns

PowerShell has a standard Verb-Noun format for command names.   
The introduction of PowerShell V2 saw a push to standardize verbs to the names published at <http://msdn.microsoft.com/en-us/library/ms714428(VS.85).aspx>

## Standard PowerShell verbs used

|  |  |
| --- | --- |
| ***Add*** | Adds a resource to a container, or attaches an item to another item. This verb is paired with the Remove verb. The resource may be a new or existing one. |
| ***Compress*** | Compacts the data of a resource. |
| ***Convert*** | Changes the data from one representation to another. |
| ***Copy*** | Copies a resource to another name or to another container. |
| ***Dismount*** | Detaches a named entity from a location. This verb is paired with the *Mount* verb. |
| ***Expand*** | Officially PowerShell defines this as "*Restores the data of a resource that has been compressed to its original state."* The library uses it for "Increase the size of". |
| ***Export*** | Encapsulates the primary input into a persistent data store, such as a file,  or into an interchange format. This verb is paired with the *Import* verb. |
| ***Get*** | Retrieves an existing resource. |
| ***Import*** | Creates a resource from data that is stored in a persistent data store (such as a file) or in an interchange format. |
| ***Invoke*** | Performs an action, such as running a command or a method. |
| ***Merge*** | Creates a single resource from multiple resources. |
| ***Mount*** | Attaches a named entity to a location. This verb is paired with the *Dismount* verb. |
| ***New*** | Officially this *Creates an empty resource that is not associated with any content.* The library uses it for creating an free standing object – for example New-VHD creates a new disk image file, without mounting it in a drive – rather than an object which is attached to something (for which *Add* is used). |
| ***Out*** | Sends data out of the environment. |
| ***Ping*** | Determines if a resource is active and if it is responding to requests. |
| ***Remove*** | Deletes a resource from a container. This verb is paired with the *Add* verb. |
| ***Rename*** | Changes the name of a resource. |
| ***Restore*** | Sets a resource to a predefined state. |
| ***Save*** | Preserves data to avoid loss. The library uses it for machine states, and it is paired with *Start*, and is an alternative to *Stop* |
| ***Select*** | Locates a resource in a container. Traditionally, e.g. in SQL, this is used to apply criteria to the container. In the library this is used when there is a need to prompt for user input to identify the resource. |
| ***Set*** | Creates a resource that contains some data, or replaces data on an existing resource. This verb is paired with the *Get* verb. |
| ***Start*** | Initiates an operation. This verb is paired with the *Stop* verb. |
| ***Stop*** | Discontinues an activity. This verb is paired with the *Start* verb. |
| ***Test*** | Verifies the operation or consistency of a resource. |
| ***Update*** | Sets a resource to a new state. |
| ***Wait*** | Pauses an operation until a specified event occurs. |

## Non-standard PowerShell verbs from version 1 of the library

In version 1 of the library some non-standard verbs were used. these were

|  |  |
| --- | --- |
| ***Apply*** | ***Apply***-Snapshot is now replaced with ***Restore-***Snapshot. |
| ***Choose*** | Was an attempt to use a different verb for "Ask for user input to identify items". ***Select*** traditionally meant only "Return a subset of data based on the provided criteria", but that definition has been expanded to encompass both kinds of selection So ***Choose*** functions have become ***Select*** ones. |
| ***Compact*** | Compact-VHD has become ***Compress***-VHD. |
| ***List*** | ***List-***VMState has become ***Get***-VMSummary. |
| ***Shutdown*** | Shutdown-VM used the integration components to start a shutdown. The command is now ***Invoke-VMShutdown.*** |
| ***UnMount*** | UnMount-VHD is replaced with the preferred verb ***Dismount.*** |

To avoid breaking scripts based on the old library, PowerShell *Aliases* have been provided to map the new names to the old ones, and these are found in the file **V1 compatibility.ps1.**

If you do not need this compatibility you may safely remove the reference to it from the module manifest **HyperV.psd1**.

# PowerShell Parameters

PowerShell allows parameters to be passed by name and by position.

Set-VMMemory "Tenby" 768MB

Is the same as

Set-VMMemory –VM "Tenby" –memory 768MB

(PowerShell understands suffixes KB, MB, GB, TB when writing numbers.)

Generally commands return an *object*, for example Get-VM "Tenby" returns a WMI object representing a virtual machine with the display name of "Tenby".

These objects can be stored in PowerShell *variables* or *piped* into other commands; when piping the parameters must be named.

The following commands all do the same thing

Get-VM "tenby" | Set-VMMemory –memory 768MB

and

$MyVm = Get-VM "tenby"  
Set-VMMemory $myVM 768MB

One general design principle is commands will either accept an object (the WMI object for the VM), or accept the information which allows the function to get the object (the name of the VM).

A second principle in PowerShell is that where it is practical to process multiple items, they can be passed as a comma separated list; for example  
Set-VMMemory "Tenby","Oxford" 1GB   
will set the memory on two VMs.

Note that a common mistake in PowerShell (especially among those new to it) would be to enter parameters which should be distinct, separated with commas, for example  
 Set-VMmemory "Tenby",1GB   
- this will assume that you want to set the memory on two VMs named "Tenby" and "1GB" but haven’t said what you want the memory to be.

In many of the functions, parameters support aliases, for example:  
 Get-VM takes a -name parameter which has an alias of "VMName".   
Parameter names can be shortened provided they are *not ambiguous* (-V might mean –Verbose or ‑VMName), but –VM can only mean VMName.

Many built-in PowerShell commands support standard parameters –whatif (which runs the command without applying any changes) –confirm (which runs the command and prompts before applying changes) –verbose (which generates additional output) and –force (which suppresses any prompt) – these are widely supported in the Module.

# Summary of functions provided

Although the library uses 31 different verbs and 69 different nouns (2,139 possible combinations), in practice many nouns are only used with one verb, and some verbs are used with only one noun, with the result that there are 126 commands, and 20 main nouns and 6 main verbs account for most of the library.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Total Verbs** | **Add** | **Get** | **New** | **Remove** | **Select** | **Set** |
| VHD | 9 |  | X | X |  |  |  |
| VM | 12 |  | X | X | X | X | X |
| VMCPUCount | 2 |  | X |  |  |  | X |
| VMDisk | 3 | X | X |  |  |  | X |
| VMDrive | 2 | X |  |  | X |  |  |
| VMFloppyDisk | 3 | X | X |  | X |  |  |
| VMHost | 2 |  | X |  |  |  | X |
| VMIntegrationComponent | 2 |  | X |  |  |  | X |
| VMKVP | 3 | X | X |  | X |  |  |
| VMLiveMigrationNetwork | 2 |  | X |  |  | X |  |
| VMMemory | 2 |  | X |  |  |  | X |
| VMNIC | 4 | X | X |  | X | X |  |
| VMNICVLAN | 2 |  | X |  |  |  | X |
| VMRASD | 4 | X |  | X | X |  | X |
| VMRemoteFXController | 4 | X | X |  | X |  | X |
| VMSCSIController | 2 | X |  |  | X |  |  |
| VMSerialPort | 2 |  | X |  |  |  | X |
| VMSnapshot | 7 |  | X | X | X | X |  |
| VMState | 2 |  |  |  |  |  | X |
| VMSwitch | 3 |  | X |  | X | X |  |
| Total | **125** | **11** | **33** | **11** | **11** | **12** | **13** |

In addition there are 16 aliases for backwards compatibility.

# Commands in the library, grouped by function

## Virtual Machine operations

|  |  |
| --- | --- |
| Start-VM | Puts the specified Hyper-V Virtual Machine into the *Running* state. |
| Stop-VM | Puts the specified Hyper-V Virtual Machine into the *Stopped* state. |
| Save-VM | Puts the specified Hyper-V Virtual Machine into the *Suspended* (saved) state. |
| Set-VMState | Sets the state of a given VM to *Running, Stopped, Suspended* etc. |
| Invoke-VMShutdown | Instructs the OS in specified Virtual Machine to begin an orderly shutdown. |
| Get-VM | Returns WMI objects representing Hyper-V Virtual Machines. |
| Select-VM | Allows the user to select a Virtual Machine from a list. |
| Get-VMSummary | Returns summary information about running VMs. |
| Get-VMThumbnail | Creates a JPG image of a running VM. |
| Get-VMBuildScript | Outputs a script which will build the VM. |
| New-VM | Creates a new Hyper-V Virtual Machine. |
| Remove-VM | Removes the specified VM from the server. |
| Get-VMSettingData | Gets the *SettingData* object(s) for one or more Virtual Machines. |
| Set-VM | Sets Name, Notes, Boot-order, Start-up, Shutdown and Recovery options. |
| Set-VMRASD | Modifies VM hardware described by Resource Allocation Setting Data. |
| Remove-VMRASD | Removes VM hardware described by Resource Allocation Setting Data. |
| Add-VMRASD | Adds an item of VM hardware described by Resource Allocation Setting Data. |
| New-VMRASD | Creates a Resource Allocation Setting Data object. |
| Ping-VM | Pings a Virtual Machine with ICMP Ping. |
| Test-VmHeartbeat | Tests the Heartbeat Integration Component to see if the VM is alive. |
| Import-VM | Imports a Virtual Machine which was previously exported. |
| Export-Vm | Exports a VM. |
| New-VmConnectSession | Opens a VMConnect session to the specified VM on the specified server. |

## VM Memory

|  |  |  |
| --- | --- | --- |
| Set-VMMemory | Sets the Memory resources allocated to one or more Virtual Machines. |  |
| Get-VMMemory | Gets the Memory resources allocated to one or more Virtual Machines. |  |

## VM Processors

|  |  |
| --- | --- |
| Get-VMCPUCount | Gets the CPU resources allocated to one or more Virtual Machines. |
| Set-VMCPUCount | Sets the CPU resources allocated to one or more Virtual Machines. |
| Get-VMProcessor | Gets Virtual CPU devices in use on one or more running Virtual Machines. |

## VM Serial Ports

|  |  |
| --- | --- |
| Get-VMSerialPort | Gets the Virtual Serial Ports on one or more Virtual Machines. |
| Set-VMSerialPort | Connects a Serial Port on a Virtual Machine to a Named Pipe. |

## VM Remote FX Controllers

|  |  |
| --- | --- |
| Add-VMRemoteFxController | Adds a Remote FX controller to one or more Virtual Machines. |
| Get-VMRemoteFxContoller | Gets the Remote FX controller(s) on one or more Virtual Machines |
| Set-VMRemoteFxController | Sets monitor count &Resolution for one or more Remote FX controllers |
| Remove-VMRemoteFxController | Removes Remote FX controller(s) from one or more Virtual Machines. |

## VM floppy disk, hard disk and DVD drives and controllers

|  |  |
| --- | --- |
| Get-VMFloppyDisk | Returns the floppy disk(s) attached to one or more VMs. |
| Add-VMFloppyDisk | Adds a floppy disk to a VM. |
| Remove-VMFloppyDisk | Removes floppy(s) disk attached to one or more VM(s). |
| Add-VMSCSIController | Adds a Synthetic SCSI controller to one or more VM(s). |
| Remove-VMSCSIcontroller | Removes a SCSI controller from one or more Virtual Machine(s). |
| Get-VMDiskController | Returns disk controllers attached to a VM. |
| Get-VMDriveByController | Returns the drives attached to a given disk controller. |
| Add-VMDrive | Adds a drive to a controller on the specified VM. |
| Remove-VMdrive | Removes a drive and/or the disk image in it from a VM. |
| Get-VMDiskByDrive | Returns the disk mounted in a given drive. |
| Get-VMDisk | Returns all disks attached to one or more VMs. |
| Add-VMDisk | Adds a disk image to a VM, mounting it in a drive. |
| Set-VMDisk | Changes the disk mounted in a drive on a VM. |
| Add-VMPassThrough | Connects a Passthrough disk to a VM. |
| Select-VMPhysicalDisk | Allows the user to choose a disk to become a pass-through disk. |
| Add-VMNewHardDisk | Creates and attaches a new virtual hard disk in one command. |

## Virtual Hard Disk and Floppy disk images

|  |  |
| --- | --- |
| New-VFD | Creates one or more Virtual Floppy Disk file(s). |
| New-VHD | Creates one or more Virtual Hard Disk file(s). |
| Get-VHDDefaultPath | Gets the default path for Virtual Hard Disk (VHD) files. |
| Get-VHD | Gets VHD files from a specified folder on a server. |
| Get-VHDInfo | Gets detailed information about one or more VHD files. |
| Get-VHDMountPoint | Returns the mount point for a VHD file, if it is mounted. |
| Mount-VHD | Mounts a VHD file to make it appear to be a disk in the parent partition. |
| Dismount-VHD | Dismounts a previously mounted VHD. |
| Expand-VHD | Increases the size of a VHD. |
| Connect-VHDParent | Reattaches a differencing VHD to its parent. |
| Merge-VHD | Merges VHDs [from snapshots]. |
| Test-VHD | Tests the working state of a disk - for example whether its parent is accessible. |
| Compress-VHD | Compacts one or more dynamic VHD file(s). |
| Convert-VHD | Creates a new VHD of a different type based on an existing VHD. |

## VM to host integration

|  |  |
| --- | --- |
| Get-VMIntegrationComponent | Gets the Integration Component data for one or more VM(s). |
| Set-VMIntegrationComponent | Enables or disables Integration Components on one or more VM(s). |
| Get-VMKVP | Gets Key/Value pairs for one or more VM(s). |
| Add-VMKVP | Adds Key/Value pairs to be sent to one or more VM(s). |
| Remove-VMKVP | Removes a Key/Value pair from the set sent to one or more VM(s) |

## VM Ethernet cards

|  |  |
| --- | --- |
| Get-VmNic | Returns information about a VM’s Network Interface Card(s). |
| Select-VmNic | Allows the user to select a Network Interface Card connected to a VM. |
| Add-VmNic | Creates a new legacy or VM-bus Network Interface Card on a VM. |
| Remove-VMNic | Removes a Network Interface Card from a VM. |
| Set-VMNICAddress | Changes the MAC address of a Network Interface Card. |
| Get-VMNicPort | Returns the switch port object for a Network Interface Card. |
| Get-VMNicSwitch | Returns the switch connected to a Network Interface Card. |
| Set-VMNICSwitch | Connects a Network Interface Card to a Virtual Switch. |
| Get-VMNicVlan | Gets the VLAN ID associated with a Network Interface Card. |
| Set-VMNicVlan | Sets the VLAN ID associated with a Network Interface Card. |
| Get-VMByMACaddress | Discovers which VM owns a particular MAC address, |

## Virtual Network Switches

|  |  |
| --- | --- |
| Select-VMExternalEthernet | Selects an available host network Interface. |
| New-VMPrivateSwitch | Creates a Virtual Network Switch, without a NIC in the parent partition. |
| New-VMInternalSwitch | Creates a Virtual Network Switch, bound to a virtual NIC in the parent partition. |
| New-VMExternalSwitch | Creates a Virtual Network Switch, bound to a physical NIC. |
| Remove-VMSwitchNIC | Removes the parent partition NIC associated with a Virtual Switch. |
| Remove-VMSwitch | Deletes a Virtual Network Switch. |
| Get-VMSwitch | Returns one or more Virtual Nework Switch objects. |
| Select-VMSwitch | Returns one or more Virtual Network Switch object(s) selected by the user. |
| New-VMSwitchPort | Creates a new port on a virtual network switch. |

## Hyper-V Servers.

|  |  |
| --- | --- |
| Set-VMHost | Configures the settings for the Hyper-V service on the host server. |
| Get-VMHost | Lists Hyper-V servers registered with Active Directory. |

## Snapshots

|  |  |
| --- | --- |
| Get-VMSnapshot | Returns Snapshots for one or more VM(s). |
| Get-VMSnapshotTree | Displays all the Snapshots of a VM in a tree view. |
| Select-VMSnapshot | Allows the user to select a Snapshot for a VM. |
| New-VMSnapshot | Creates a new Snapshot of one more VM(s). |
| Rename-VMsnapshot | Changes the display name of a Snapshot. |
| Update-VMSnapshot | Replaces an existing Snapshot with a new one. |
| Restore-VMsnapshot | Rolls a VM back (or forward) to a snapshot. |
| Remove-VMSnapshot | Removes one or more Snapshot(s) from a VM. |

## Cluster Operations

|  |  |
| --- | --- |
| Move-VM | Moves one or more VM(s) between Cluster Nodes using Live Migration. |
| Select-ClusterSharedVolume | Allows the user to select a Clustered Shared Volume from a list. |
| Sync-VMClusterConfig | Synchronizes the configuration of a VM across all members of a Cluster. |
| Get-VMClustergroup | Returns the Cluster Group of which the VM is a member. |
| Select-VMLiveMigrationNetwork | Allows the user to select Cluster Networks to be used for Live Migration. |
| Get-VMLiveMigrationNetwork | Returns a list of Cluster Networks in use for Live Migrating VMs. |

## User Interface

|  |  |  |
| --- | --- | --- |
| Show-HypervMenu | Displays a menu to manage Hyper-V. |  |
| Show-VMMenu | Displays a menu to manage an individual VM. |  |
| Show-VMDiskMenu | Displays a menu to manage an individual VM's disks. |  |
| Show-VHDMenu | Displays a menu to manage Virtual Hard Disks. |  |
| Out-Tree | Outputs objects in a tree format. |  |
| Select-Tree | Allows the user to select from objects in a tree format. |  |
| Select-List | Returns an object selected by the user from a table of numbered rows. |  |
| Select-Item | Returns a zero-based integer indicating the user's selection. |  |

## Miscellaneous Admin functions

|  |  |
| --- | --- |
| Test-WMIJob | Checks the status of background WMI jobs. |
| Test-WMIResult | Checks the result returned by calling a WMI method. |
| Convert-DiskIDtoDrive | Converts a logical disk index to a drive letter. |
| Get-FirstAvailableDriveLetter | Returns a CHAR indicating the first available drive letter. |
| Wait-ForDisk | Waits for disk to come on line. |
| Test-Admin | Checks to see if the current session has administrator privileges. |
| ConvertTo-Enum | Converts Hash Table data to an enumeration type definition. |
| Select-Enumtype | Returns a value selected by the user from an enumeration type. |
| New-Zip | Creates a new , empty .Zip file. |
| Get-ZIPContent | Returns information about the contents of a .Zip file. |
| Add-ZIPContent | Adds content to a .Zip file. |
| Copy-ZipContent | Copies content out of a .Zip file. |

# Detailed Explanation of the functions

## Start-VM

Puts the specified Hyper-V Virtual Machine into the Running state.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to start. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |
| **Wait** |  | Specifies that the command should not return until the WMI job is complete. This does not guarantee that the Operating system in the VM is responsive. To check that the OS has loaded use the HeartBeatTimeout Parameter. |
| **HeartBeatTimeOut** |  | If specified the VM is checked for a response from the Heartbeat integration component every 5 seconds until either one is found or the timeout expires. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and Recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Starting a VM restores the contents of memory if they were saved. A background WMI job is created to change the state.

### Example

Start-VM "London-DC" -HeartBeatTimeOut 300

Starts the VM named "London-DC" and waits up to 5 minutes for a heartbeat before running the next command.

### Notes

Start-VM calls Set-VMstate passing it the VM and server it received..

### See Also:

Set-VMState

Stop-VM

## Stop-VM

Puts the specified Hyper-V Virtual Machine into the Stopped state.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to Stop. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |
| **Wait** |  | Specifies that the command should not return until the WMI job is complete. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and Recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none"). |

### Description

Stopping a VM powers it off without saving its state. If it is in a saved state, the saved state information is deleted. A background WMI job is created to change the state.

### Examples

Stop-VM (Select-VM -Server "JAMES-2008" -Multiple)

Prompts the user to select one or more of the VMs on the server named "JAMES-2008", and stops the selected VMs.

Get-VM -running | Stop-VM

Stop all VMs that are in the Running state.

### Notes

Stop-VM calls Set-VMstate passing it the VM and Server parameters it received.

### See Also:

Set-VMState

Start-VM

## Save-VM

Puts the specified Hyper-V Virtual Machine into the Suspended (saved) state.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to save. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |
| **Wait** |  | Specifies that the command should not return until the WMI job is complete. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and Recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none"). |

### Description

Saving a VM stores the contents of memory before powering it off. A background WMI job is created to change the state.

### Examples

Stop-VM (Select-VM -Server "JAMES-2008" -Multiple)

Prompts the user to select one or more of the VMs on the server named "JAMES-2008", and saves the selected VM(s).

Get-VM -running | Save-VM

Saves all VMs that are in the Running state.

### Notes

Save-VM has an alias of suspend-VM. It calls Set-VMstate passing it the VM and Server parameters it received..

### See Also:

Set-VMState

## Set-VMState

Sets the state of one or more VM(s) to Running, Stopped, Suspended etc.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) whose state should be changed. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **State** *Required* |  | The desired new state If a valid integer or state name is passed, it is converted to the VMState Enum. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and Recursive calls to the command. |
| **Wait** |  | Specifies the command should not return until the WMI job it starts has completed. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Sets the state of one or more VM(s) to Running, Stopped, Suspended etc.

### Examples

Get-VM -Name "Core-%" | Set-VMState -State [VMState]::Running

Starts all VMs with names that start with "CORE-". using the [VMState] enum.

Set-VMState -VM "Core","Tenby" -Server "JAMES-2008" - State "Running"

Starts the VMs named "Core" and "Tenby" on the server "JAMES-2008" to the running state using the name of the state.

### Notes

VMState is intended for use by Start-, Stop- and Save- VM, but may be called directly.

### See Also:

[RequestStateChange Method on MSDN](http://msdn.microsoft.com/en-us/library/cc723874(VS.85).aspx)

## Invoke-VMShutdown

Instructs the OS in specified Virtual Machine to begin an orderly shutdown.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to modify. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Reason** |  | Sent to the guest OS for information about why the remote shutdown was started. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |
| **Wait** |  | Specifies that the command should not return until the WMI job is complete. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and Recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none".). |

### Description

Stopping a VM does not close down the operating system inside it. The Hyper-V integration components include a Shutdown component, which tells the guest OS to start a clean shut down. This command invokes that component - and will only work against VMs which have it installed.

### Examples

Invoke-VMShutdown (Select-VM -Server "JAMES-2008" -Multiple) -force

Prompts the user to select one or more of the VMs on the server "JAMES-2008", and shuts down the selected VMs without further prompting.

Get-VM -Running | Invoke-VMShutdown

Shuts down all VMs that are in the Running state on the local server.

### See Also:

[ShutdownComponent - InitiateShutdown Method on MSDN](http://msdn.microsoft.com/en-us/library/cc703625(VS.85).aspx)

## Ping-VM

Pings a Virtual Machine with ICMP Ping.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to check. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |

### Description

First attempts to use integration components to discover the Fully Qualified Domain Name of the VM. Then attempts to ping that FQDN using ICMP from the machine where the command is running.

### Examples

Ping-VM "Tenby" -server james2008

Attempts to ping from the local machine to the VM named "Tenby" on the server James-2008.  
This relies on the integration components being present and the FQDN they return being resolvable on the local machine. .

get-VM -r | foreach-object {if ((Ping-VM $\_).statusCode -ne 0) {"$($\_.elementname) is inaccessible"} }

Gets the running VMs on the local server, pings each of them and outputs a message for any which are running but can't be pinged.

### See Also:

Get-VMKvp

[WIn32\_PingStatus WMI object on MSDN](http://msdn.microsoft.com/en-us/library/aa394350(VS.85).aspx)

## Test-VMHeartbeat

Tests the HeartBeat integration component, to see if the VM is alive.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to check. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |
| **HeartBeatTimeOut** |  | If specified the VM, is checked for a response from the heartbeat integration component every 5 seconds until one is found or the timeout expires. |

### Description

Although not a 100% reliable test because the OS in the VM may not have integration components available, they may not be installed or may be disabled. But where they are installed the provided the best indication of VM state.

### Examples

start-VM "London DC" ; Test-VMheartBeat "London DC" -Timeout 300; start-VM "London SQL"

Starts the VM named "London DC" and waits up to 5 minutes for its heartbeat. (This is the same as calling Start-VM "London-DC" -HeartBeatTimeOut 300. ) Then starts the VM "London SQL".

### See Also:

[MSVM\_HeartbeatComponentClass on MSDN](http://msdn.microsoft.com/en-us/library/cc136840(VS.85).aspx)

## Get-VM

Returns WMI objects representing Hyper-V Virtual Machines.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Name** | (Input from pipeline) Wildcards | Specifies the name (or name pattern) to look for. WMI wild cards use % , but \* will be converted to % to allow familiar style to be used. Can be used with an Alias of VMName (which may be shortened to VM). |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |
| **Suspended** |  | Only Return Virtual Machines that are in the SUSPENDED state. |
| **Running** |  | Only return Virtual Machines that are in the RUNNING state. |
| **Stopped** |  | Only return Virtual Machines that are in the STOPPED state. |

### Description

Gets Virtual Machine objects on one or more Hyper-V servers; VMs may be selected based on their name or state (running, stopped etc.).

### Examples

Get-VM

Returns WMI MSVM\_ComputerSystem objects for all Virtual Machines on the local server (Note, the parent partition is filtered out).

Get-VM -Name "Windows 2008 Ent Full TS"

Returns a single WMI MSVM\_ComputerSystem object for the VM named "Server 2008 ENT Full TS".

Get-VM -Name "%2008% -Server "JAMES-2008"

Returns WMI MSVM\_ComputerSystem objects for the VMs with "2008" in their names from the server named JAMES-2008.

### See Also:

New-VM

Remove-VM

Set-VM

[MsVM\_ComputerSystem Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136822(VS.85).aspx)

## Select-VM

Allows the user to select a Virtual Machine from a list.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Server** |  | Specifies the Hyper-V server on which Virtual Machines reside. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |
| **Multiple** |  | If specified allows multiple VMs to be selected from the list. If omitted only a single VM may be selected. |

### Description

If only one VM is available then it is returned; otherwise a list of VMs is displayed and the user is prompted to make a selection.

### Examples

Select-VM -multiple

Lets the user select one or more VMs from a list of those on the local machine.

Select-VM -Server James-2008,Jackie-2008

Lets the user select a single VM from a list of those on the cluster nodes named "James-2008" and "Jackie-2008".

## Get-VMByMACaddress

Discovers which VM owns a particular mac address.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Mac** *Required* | (Input from pipeline) | The MAC address to seek. The command uses regular expression matching, so a partial address or regex can be used in the MAC field. |
| **Server** |  | Specifies the Hyper-V server(s) to search for a matching NIC By default "." , the local computer is used. An array of server names by be passed but wildcards cannot be used. |

### Examples

Get-VMbymacAddress "00155D000101"

Returns details of the VM with the NIC given the address 00155D000101.

get-VM (get-VMbyMacAddress "00155DD0BEEF").VM

Returns the WMI object representing that VM.

## New-VMConnectSession

Opens a VMConnect session to the specified VM on the specified server.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to connect to. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |

### Description

VMConnect uses RDP to connect to a "Virtual KVM switch" on the Virtualized graphics card, keyboard and mouse of the Virtual Machine, rather than to any Remote Desktop Service running in its Operating System. The connection is made by invoking %programFiles%\hyper-v\VMConnect.exe. The same program which is called by the GUI Hyper-V manager.

### Examples

New-VMConnectSession -VM "tenby"

Launches a VMConnect session to a Virtual Machine named "Tenby" on the local server.

New-VMConnectSession -VM $tenby -server James-2008

Launches a VMConnect session to the VM described by the $tenby variable, located on the server James-2008.

$VMConnectPID = New-VMConnectSession -VM "tenby" ; Start-Sleep -Seconds 60 ; Stop-Process -Id $VMConnectPID

Launches a VMConnect session to the VM named "Tenby" on the local server, waits one minute, and then closes VMConnect.

## Get-VMThumbnail

Creates a PNG image of a running VM.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to capture a picture of. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **width** |  | The width of the Image in Pixels (default 800). |
| **Height** |  | The Height of the Image in Pixels (default 600). |
| **Path** |  | The path to save the bitmap image to. If no name is specified the VM name is used. If no directory is included the current one is used. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. |
| **Passthru** |  | Specifies that the Bitmap image should be returned as an object and not saved as a file. |

### Description

Creates a PNG image of a running VM, at a given width and height. The image can either be saved to a file or passed on to another command.

### Examples

Get-VMThumbnail core

Gets a 800x600 Pixel PNG image for the machine named core, and writes it as core.PNG in the current folder..

While ($true) { Get-VMThumbnail -VM "core" -w 640 -h 480 ` -path ((get-date).toLongTimeString().replace(":","-") + ".PNG") Sleep -Seconds 10 }

Creates a loop which continues until interrupted; in the loop, creates an image of the VM "Core" with a file name based on the current time. Then waits 10 seconds and repeats.

Get-VM -Running -server "James-2008" | Get-VMThumbnail -w 320 -h 240 -path images

Discovers running VMs on Server named "James-2008", and writes a 320x240 size image to the folder named images for each one.

### See Also:

[GetVirtualSystemThumbnailImage method on MSDN :](http://msdn.microsoft.com/en-us/library/cc160707(VS.85).aspx)

## Get-VMSummary

Returns summary information about running VMs.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to provide information about. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified a summary is returned for all VMs. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |

### Description

For each VM a custom object is returned with the following properties Host The Hyper-V server hosting the VM VMElementName The Display Name for the VM Name The GUID which uniquely identifies the VM CreationTime The Time the VM was created EnabledState The current state of the VM as a Enum Notes The Notes field for the VM CPUCount The number of CPUs assigned to the VM CPULoad The instantaneous CPU load on the VM CPULoadHistory An Array of CPU loads for the last 100 seconds MemoryUsage Memory used by the VM Heartbeat The State of the VM heartbeat as a number HeartbeatText The State of the VM heartbeat as converted to text Uptime The VM uptime in Milliseconds UptimeFormatted The VM uptime formatted as Hours, minutes and seconds GuestOS The Guest OS Identity string returned by KVP exchange Snapshots The number of Snapshots which exist of the Virtual Machine Jobs The asynchronous jobs associated with the VM FQDN The fully qualified domain name returned by KVP exchange IpAddress The IP address found when attempting to ping the FQDN.

### Examples

Get-VMSummary -server james-2008,jackie-2008 | format-table -autosize

Outputs formatted status for all VMs on the servers named "James-2008" and "Jackie-2008" (subject to the constraints of screen width).

Get-VMSmmary "Windows 2008 Ent Full TS"

Outputs status for the VM named "Server 2008 ENT Full TS" on the local server.

## Get-VMBuildScript

Outputs a script which will build the VM.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** | (Input from pipeline) Wildcards | The Virtual Machine(s) to get build scripts for. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified the script is output for all VMs.7. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |

### Description

Outputs New-VM, add-VMxxxx, Set-VMxxx commands to recreate the same VM.

### Examples

Get-VMBuildScript "Tenby"

Outputs a script to build the VM named "Tenby".

## Get-VMSnapshot

Returns SnapShots for one or more VM(s).

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** | (Input from pipeline) Wildcards | The Virtual Machine(s) to query . This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is passed, all VMs on the server are queried. |
| **Name** |  | The name of a snapshot to return. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **Newest** |  | If Specified only the newest Snapshot is returned. |
| **Root** |  | If specified only the root snapshot is returned. |
| **Current** |  | If specified only the current snapshot is returned. |

### Description

Returns one or more MSVM\_VirtualSystemSettingData WMI object(s) representing snapshots. An individual snapshot can be selected by specifying its name, and the newest snapshot can be selected by specifying the -Newest switch.

### Examples

Get-VMsnapshot $Core -newest

Returns the newest snapshot on the VM pointed to by $core.

### See Also:

Get-VMsnapshotTree

New-VMsnapshot

Remove-VMSnapshot

Rename-VMSnapShot

Restore-VMSnapShot

Select-VMSnapshot

## Get-VMSnapshotTree

Displays all the Snapshots of a VM in a tree view.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine to query. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), It may be passed via the pipe, but must be a single item. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |

### Description

The function displays the snapshots of a Single Virtual machine, arranged in tree format to make it easy to see which snapshots are descended from which others.

### See Also:

Get-VMSnapshot

Select-VMSnapshot

## New-VMSnapshot

Creates a new snapshot of one more VMs.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to snapshot. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Note** |  | Sets an explanatory note on the snapshot. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **Wait** |  | If the wait switch is specified the function will display a progress indicator until the job completes. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none"). |

### Description

Creates a new snapshot of one more VMs.

### Examples

new-VMsnapshot $Core

Takes a snapshot of the VM pointed to by $core.

get-VM "core%" -server "James-2008" | new-VMSnapshot -wait

Gets the VMs with names beginning "Core" on the server "James-2008" and snapshots them one by one.

### See Also:

Get-VMsnapshot

Get-VMsnapshotTree

Remove-VMSnapshot

Rename-VMSnapShot

Restore-VMSnapShot

Select-VMSnapshot

[CreateVirtualSystemSnapshot Method of the MsVM\_VirtualSystemManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136785(VS.85).aspx)

## Remove-VMSnapshot

Removes one or more snapshots from a VM.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Snapshot** *Required* | (Input from pipeline) | An Object representing the snapshot to be removed - may be passed via the pipeline.. |
| **Tree** |  | If specified remove-VMSnapshot will remove snapshots which are children of the selected one.. |
| **Wait** |  | If the wait switch is specified the function will display a progress indicator until the job completes. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none"). |

### Description

Removes one or more snapshots from a VM.

### Examples

Select-VMsnapshot $Core | remove-VMsnapshot -tree

Prompts the user to select a snapshot on the VM pointed to by $core, and removes it, and any child snapshots.

### See Also:

Get-VMsnapshot

Select-VMSnapshot

[RemoveVirtualSystemSnapshot Method of the MsVM\_VirtualSystemManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136962(VS.85).aspx)

## Rename-VMsnapshot

Changes the display name of a snapshot.

### Description

Changes the display name of a snapshot.

### Examples

Rename-VMSnapshot –v $core –s (Select-VMsnapshot $core).elementName –n "default"

Prompts the user to select one of the snap shots on the VM pointed to by $core and renames it to "default".

Select-VMsnapshot $core | Rename-VMSnapshot -newname "default"

An Improved version of Example 1.

### See Also:

Get-VMSnapshot

New-VMSnapshot

Update-VMSnapshot

## Restore-VMSnapshot

Rolls a VM back (or forward) to a snapshot.

### Parameters

|  |  |  |
| --- | --- | --- |
| **SnapShot** *Required* | (Input from pipeline) | An Object representing one or more snapshots to be applied - may be passed via the pipeline. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none"). |
| **Restart** |  | Restarts the VM(s) after completion of the restoration(s). |
| **Wait** |  | If the wait switch is specified the function will display a progress indicator until the restoration completes. |

### Description

Rolls a VM back (or forward) to a snapshot.

### Examples

Select-VMsnapshot $Core | Restore-VMsnapshot

Lets the user select a snapshot on the VM pointed to by $core and applies it.

Get-VMsnapshot -name "Monday" | Restore-VMsnapshot -force

Gets snapshots named "Monday" made on any VM on the local server and restores them without prompting.

### See Also:

Get-VMSnapshot

New-VMSnapshot

Remove-VMSnapshot

Select-VMSnapshot

[ApplyVirtualSystemSnapshot Method of the MsVM\_VirtualSystemManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136767(VS.85).aspx)

## Select-VMSnapshot

Allows the user to select a snapshot for a VM.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine to query This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches a single VM). It may be passed via the pipe. Only a single VM may be queried. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |

### Description

If there is only a single snapshot it is returned, if there is more than one a tree view of the snapshots for the VM is displayed for the user to make a selection.

### Examples

Select-VMsnapshot $Core

Gets the Snapshots of the machine pointed to by $core and if there are multiple snap shots prompts the user to select one from a tree.

### See Also:

Get-VMSnapshot

Get-VMSnapshotTree

## Update-VMSnapshot

Replaces an existing snapshot with a new one.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) on which the snapshot should be updated. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **SnapName** |  | The name of the snapshot to update. If no name is provided the newest snapshot is used (and its name re-used). |
| **Note** |  | A note to add to the newly created snapshot. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. Update-VMSnapShot calls other commands which have a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none"). |

### Description

This function is a wrapper for New-Snapshot, remove-Snapshot, and rename-snapshot. If no snapshot name is specified the function gets the most recent snapshot. It renames that snapshot to "Delete-me". It creates a new snapshot, using the note if one is specified, and renames the new snapshot to the given name. Finally it removes the old version of the snapshot..

### See Also:

Get-VMSnapshot

New-VMSnapshot

Rename-VMSnapshot

## Export-VM

Exports a VM.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to export. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Path** *Required* |  | The location where the exported VM files will be placed.. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |
| **CopyState** |  | If present indicates that the state, such as virtual hard disks, saved state files, and memory content files, should be exported. |
| **Wait** |  | Specifies that the command should not return until the WMI job is complete. |
| **Preserve** |  | Specifies that a zip file should be made containing those files which are consumed during import. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and Recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

The command supports the option to the copy the state of the VM (as in the GUI) and can also make a backup copy of the files which will be consumed during import.

### See Also:

Import-VM

[ExportVirtualSystem method on MSDN](http://msdn.microsoft.com/en-us/library/cc136793(VS.85).aspx)

## Import-VM

Imports a Virtual Machine which was previously exported.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Paths** *Required* | (Input from pipeline) | The location(s) of the exported files, the name may be written as Path. If no Path is passed as a parameter, the Import-VM looks for input to be piped to it. If an object is piped, the command will look for Fullname, Path or Diskpath properties, in that order. |
| **Server** |  | Specifies the Hyper-V server on which the import is to be performed. By default "." , the local computer, is used. Only a single server name may be specified for import. |
| **ReImportVM** |  | If specified the ZIP files for an import are copied back to their location and the VM specified is deleted. Note that no check is performed to confirm the deleted VM is the same one which will be imported. |
| **ReUseIDs** |  | Indicates the re-imported VM should use the same IDs as when it was exported. This can cause a conflict if the old VM still exists. |
| **Wait** |  | Specifies that the command should not return until the WMI job is complete. |
| **Preserve** |  | Specifies that a zip file should be made containing those files which are consumed during import.. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and Recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Imports a Virtual Machine which was previously exported.

### See Also:

Export-VM

[ImportVirtualSystem method on MSDN](http://msdn.microsoft.com/en-us/library/cc136798(VS.85).aspx)

## New-VM

Creates a new Hyper-V Virtual Machine.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Name** *Required* | (Input from pipeline) | Specifies the display name of the new Virtual Machine. Hyper-V does not require these names to be unique. |
| **Path** |  | The location for VM files, including snapshots. Hyper-V has a default location, which will be used if no path is specified. |
| **Server** |  | Specifies the Hyper-V server on which the Virtual Machine will reside. By default "." , the local computer, is used. Unlike other commands, New-VM supports only a single server. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and Recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Creates a new Hyper-V Virtual Machine, and returns a WMI object representing it. After the machine is created you should set memory, CPU count, network adapters and disks.

### Examples

New-VM -Name "Tenby" -Server "James-2008"

Creates a new VM named "Tenby" on the server named "James-2008".

### See Also:

Get-VM

Remove VM

[DefineVirtualSystem Method on MSDN](http://msdn.microsoft.com/en-us/library/cc136786(VS.85).aspx)

## Remove-VM

Removes the specified VM from the server.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to remove. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and Recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none"). |

### Description

Delete a VM and its snapshots, but does not delete its Virtual Hard Disk files.

### Examples

Remove-VM -VM "Tenby"

Removes the VM named "Tenby" from the local server.

### See Also:

New-VM

[DestroyVirtualSystem Method on MSDN](http://msdn.microsoft.com/en-us/library/cc136790(VS.85).aspx)

## Get-VMSettingData

Gets the Setting data object for one or more Virtual Machines.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to query. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified the information will be returned for all VMs on the server. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |

### Description

Returns an MSVM\_VirtualSystemSettingData object describes the boot devices for a VM, its display name and notes.

### See Also:

[MSVM\_VirtualSystemSettingData object on MSDN](http://msdn.microsoft.com/en-us/library/cc136944(VS.85).aspx)

## Set-VM

Sets name, notes, boot order, start-up, shutdown and recovery options.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to configure. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Name** |  | The new display name for the VM. |
| **BootOrder** |  | An array of enums representing boot devices. If valid integers or strings are passed they will be converted to enums. |
| **Notes** |  | The new contents of the VM Notes field. |
| **AutoRecovery** |  | An enum representing a recovery action. If valid integers or strings are passed they will be converted to enums. |
| **AutoShutdown** |  | An enum representing a Shutdown action. If valid integers or strings are passed they will be converted to enums. |
| **AutoStartUP** |  | An enum representing a start-up action. If valid integers or strings are passed they will be converted to enums. |
| **AutoDelay** |  | The number of seconds to delay an automated startup (to allow other VMs to boot first). |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and Recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Sets name, notes, boot order, start-up, shutdown and recovery options for one or more VM(s).

### Examples

Set-VM $VM -bootorder CD,IDE,net,Floppy

Sets the boot order for the machine whose config is $VM to CD, IDE, Network, Floppy.

set-VM -VM "core" -autoStart AlwaysStartup

Sets the VM named core on the local host to start whenever the host OS boots.

Set-VM "CORE-%" -bootorder CD,IDE,net,Floppy -autoStart

Take all the machines whose names begin CORE- and set their boot order and start-up action.

### See Also:

Get-VMSettingData

[MsVM\_VirtualSystemGlobalSettingData WMI object on MSDN](http://msdn.microsoft.com/en-us/library/cc136939(VS.85).aspx)

[MSVM\_VirtualSystemSettingData WMI object on MSDN](http://msdn.microsoft.com/en-us/library/cc136944(VS.85).aspx)

## Set-VMRASD

Modifies virtual hardware described by Resource Allocation Setting Data to a VM.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) | The Virtual Machine to modify, as a WMI object. |
| **RASD** *Required* |  | The Resource Allocation Settings Data WMI Object representing the updated hardware. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Exists to support other commands. Not intended to be called from the command line.

### See Also:

[ModifyVirtualSystemResources method of MSVM\_VirtualSystemManagementService on MSDN](http://msdn.microsoft.com/en-us/library/cc136807(VS.85).aspx)

## New-VMRASD

Creates a Resource Allocation Setting Data object.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Restype** *Required* |  | Resource type. |
| **ResSubType** |  | Resource Sub type. |
| **Server** *Required* |  | Specifies the Hyper-V server on which to create the object. By default "." , the local computer, is used. Only a single Server May be specified. |

### Description

Exists to support other commands. Not intended to be called from the command line.

### See Also:

[MSVM\_AllocationCapabilities on MSDN](http://msdn.microsoft.com/en-us/library/cc136817(VS.85).aspx)

[MSVM\_SettingsDefineCapabilities on MSDN](http://msdn.microsoft.com/en-us/library/cc136891(VS.85).aspx)

## Add-VMRASD

Adds virtual hardware described by Resource Allocation Setting Data to a VM.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) | The Virtual Machine to modify, as a WMI object. |
| **RASD** *Required* |  | The Resource Allocation Settings Data WMI Object representing the new hardware. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Exists to support other commands. Not intended to be called from the command line.

### See Also:

[AddVirtualSystemResources method of MSVM\_VirtualSystemManagementService on MSDN](http://msdn.microsoft.com/en-us/library/cc160705(VS.85).aspx)

## Remove-VMRASD

Removes virtual hardware described by Resource Allocation Setting Data from a VM.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) | The Virtual Machine to modify, as a WMI object. |
| **RASD** *Required* |  | The Resource Allocation Settings Data WMI Object representing the hardware to be removed. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE.It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the comman. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none". |

### Description

Exists to support other commands. Not intended to be called from the command line.

### See Also:

[RemoveVirtualSystemResources method of MSVM\_VirtualSystemManagementService on MSDN](http://msdn.microsoft.com/en-us/library/cc160711(VS.85).aspx)

## Get-VMMemory

Gets the memory resources allocated to one or more Virtual Machines.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to query. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified the information will be returned for all VMs on the server. |
| **Server** *Required* |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer,, is used. An array of server names may be passed but wildcards can not be used. |

### Description

Gets the memory resources allocated to one or more Virtual Machines.

### Examples

Get-VMMemory -VM "Tenby" -Server "JAMES-2008"

Gets the memory allocated to the VM named "Tenby" on the host "JAMES-2008".

### See Also:

Set-VMMemory

[MSVM\_MemorySettingData object on MSDN](http://msdn.microsoft.com/en-us/library/cc136856(VS.85).aspx)

## Set-VMMemory

Sets the memory resources allocated to one or more Virtual Machines.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to modify. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Memory** *Required* |  | Specifies the amount of memory that should be allocated to the Virtual Machine at start-up. For machines without dynamic memory enabled this quantity never changes. For machines with dynamic memory this is both the initial amount ("Virtual Quantity") and the Reserved amount ("Reservation"). The GUI interface can't set these to different values, and doing so is assumed to be unsupported. If greater than 2MB it is assumed to be expressed in bytes, otherwise it is treaded as MB. For clarity in scripts it has an alias MemoryInBytes. |
| **Limit** |  | This Parameter is ignored unless -Dynamic is specified. The maximum amount of memory that may be consumed by the virtual system. On a system without dynamic memory this is equal to Memory. With dynamic memory it sets the ceiling for memory allocation. If greater than 2MB it is assumed to be expressed in bytes, otherwise it is treaded as MB. |
| **Weight** |  | This Parameter is ignored unless -Dynamic is specified. Defines the memory allocation weighting value for a VM. After all reserves have been met, the remaining memory of the hosting platform will be allocated to virtual systems based on their relative weights. Prior to Server 2008-R2 SP1 this value was ignored, but had a maximum of 1,000. Server 2008-R2 SP1 uses this value with a maximum of 10,000. The default is 1,000 to avoid errors when used with older systems. For VMs using dynamic memory it is highly recommended that you specify the weight explicitly.. |
| **BufferPercentage** |  | This Parameter is ignored unless -Dynamic is specified. For a VM with dynamic memory enabled, defines the amount of extra memory that should be reserved at runtime, as a percentage of the total memory that the Virtual Machine is thought to need. This property can be in the range of 5 to 2000, the default is 20 (as it is in the GUI). |
| **Dynamic** |  | Specifies that the VM should use the dynamic memory feature. This is not available in Server 2008, or Server 2008-R2 before SP1. |
| **Server** *Required* |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Sets the memory resources allocated to one or more Virtual Machines. Dynamic memory was introduced in Server 2008-R2 SP1, and is managed with the -Limit, -Weight, -BufferPecentage and -Dynamic options.

### Examples

Get-VM -Name "Core-%" | Set-VMMemory -Memory 1073741824

Allocates 1GB of memory to all of the VMs on the local server with a name that starts with "CORE-".

Set-VMMemory -VM "Tenby" -Memory 1.5GB -Server James-2008

Allocates 1.5GB of memory to the VM named Tenby on the server JAMES-2008.

get-VM server\* | Set-VMMemory -Limit 2GB -buffer 25 -weight 8000 -dynamic

Enables Dynamic memory on all machines whose names start with "Server". Setting the upper limit of memory to 2GB, a 25% buffer and a relative weight of 8000.

### See Also:

Get-VMMemory

[MSVM\_MemorySettingData object on MSDN](http://msdn.microsoft.com/en-us/library/cc136856(VS.85).aspx)

## Get-VMCPUCount

Gets the CPU Resources allocated to one or more Virtual Machines.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to query. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified the information will be returned for all VMs on the server. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |

### Description

Gets the CPU Resources allocated to one or more Virtual Machines.

### Examples

Get-VMCPUCount core

Returns the CPU settings for the VM named core on the local server..

### See Also:

Set-VMCPUCount

[MsVM\_ProcessorSettingData object on MSDN](http://msdn.microsoft.com/en-us/library/cc136870(VS.85).aspx)

## Set-VMCPUCount

Sets the CPU Resources allocated to one or more Virtual Machines.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to modify. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **CPUCount** |  | The Number of Virtual Processors to assign to the VM. It must be in the range 1-4. It can not be greater than the number of physical cores in the host. Some OSes support fewer processors.. |
| **Limit** |  | The maximum time this processor is allowed to use. A VM defaults to a limit of 100% unless configured otherwise. |
| **Reservation** |  | Amount of CPU time reserved for a VM. A VM defaults to a reservation of 0% unless configured otherwise. |
| **Weight** |  | Relative Weighting used to assign CPU time to this VM when the host processor is fully utilized. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Sets the CPU Resources allocated to one or more Virtual Machines.

### Examples

Set-VMCPUCount "tenby" 2 -Server "James-2008"

Assigns 2 CPUs to the VM named Tenby on Server James-2008.

Get-VM Core-% | Set-VMCPUCount -CPUCount 2

Gives 2 CPUs to all VMs on the local machine whose names begin CORE-.

### See Also:

Get-VMCpuCount

[MsVM\_ProcessorSettingData object on MSDN](http://msdn.microsoft.com/en-us/library/cc136870(VS.85).aspx)

## Get-VMKVP

Gets Key/Value pairs for one or more Virtual Machines.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to query. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified the information will be returned for all VMs on the server. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |

### Description

Hyper-V provides an Integration Component to move a small amount of data between the registry in a guest VM and the host. This data is in the form of Keys (names) and associated Values..

### Examples

(Get-VMKVP "Windows 2008 Ent Full TS").OSName

Returns "Windows Server (R) 2008 Enterprise" - the OS that VM is running.

Get-VMkvp % -server james-2008

Returns the Key/Value pairs sent back by all the VMs on the Server James-2008.

Get-VM -running | Get-VMKVP

Returns the Key/Value pairs for running VMs on the local Server.

### Notes

The values sent Automatically to the child VM can be found in HKLM:\SOFTWARE\Microsoft\Virtual Machine\guest\Parameters.

The values sent Programmatically to the child VM can be found in HKLM:\SOFTWARE\Microsoft\Virtual Machine\External.

Those sent by the Child VM are in HKLM:\SOFTWARE\Microsoft\Virtual Machine\auto.

If the VM isn't running its Key/Value Pair Exchange Service does NOT persist the values. So stopped VMs won't return anything !.

### See Also:

Add-VMKVP

[MsVM\_KvpExchangeComponent on MSDN](http://msdn.microsoft.com/en-us/library/cc136848(VS.85).aspx)

## Get-VMProcessor

Gets the virtual CPU devices connected to one or more Virtual Machines.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to query. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified the information will be returned for all VMs on the server. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |

### Description

Get-VMCPUCount returns the number of virtual processors assigned to a VM: Get-VMProcessor returns the objects representing those processors when the VM is running..

### Examples

Get-VMProcessor "core"

Returns the Virtual CPU objects for the VM named core on the local server..

### See Also:

Get-VMCPUCount

[MsVM\_Processor object on MSDN](http://msdn.microsoft.com/en-us/library/cc136866(VS.85).aspx)

## Get-VMSerialPort

Gets the virtual Serial port devices connected to one or more Virtual Machines.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to query. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified the information will be returned for all VMs on the server. |
| **PortNumber** |  | The Serial Port number, either 1 or 2. If not specified both serial ports will be returned. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |

### Description

Gets the virtual Serial port devices connected to one or more Virtual Machines.

### Examples

Get-VMSerialPort "core"

Gets the settings for both seral ports of the VM named "Core" on the local server.

### See Also:

Set-VMSerialPort

[MsVM\_SerialPort Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136887(VS.85).aspx)

## Set-VMSerialPort

Connects a serial port on the specified Hyper-V Virtual Machine to a named pipe.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to modify. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **PortNumber** |  | The port to connect to (1 or 2). |
| **Connection** |  | The named-pipe path to the connect the port to. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Connects a serial port on the specified Hyper-V Virtual Machine to a named pipe.

### Examples

Set-VMSerialPort "CORE" 2 "\\.\PIPE\WIBBLE"

Connects serial port 2 on the VM "Tenby" on the local server.

### See Also:

Get-VMSerialPort

[MsVM\_SerialPort Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136887(VS.85).aspx)

## Get-VMIntegrationComponent

Gets the Integration Component data for one or more Virtual Machines.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to query. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified the information will be returned for all VMs on the server. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |

### Description

Returns Resource Allocation Setting Data objects describing the Integration Components for a VM.

### See Also:

Set-VMIntegrationComponent

[Integration Components Classes on MSDN](http://msdn.microsoft.com/en-us/library/cc136801(VS.85).aspx)

## Set-VMIntegrationComponent

Enables or disables Integration Components on one or more Virtual Machines.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to modify. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **ComponentName** |  | The Name(s) of the Integration Component(s) to be modified. Only the start of the name needs to specified as a Wildcard is applied to the end of the name. |
| **State** |  | The Text "Running" or "Stopped" or the values 0 or 1 may be passed. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Enables or disables Integration Components on one or more Virtual Machines.

### See Also:

Get-VMIntegrationComponent

[Integration Components Classes on MSDN](http://msdn.microsoft.com/en-us/library/cc136801(VS.85).aspx)

## Add-VMKVP

Adds Key/Value pairs to be sent to one or more Virtual Machines.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to modify. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Key** *Required* |  | The Name of the key. |
| **Value** *Required* |  | The Value Associated with the key. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Adds Key/Value pairs to be sent to one or more Virtual Machines.

### Notes

The values sent Automatically to the child VM can be found in HKLM:\SOFTWARE\Microsoft\Virtual Machine\guest\Parameters.

The values sent Programmatically to the child VM can be found in HKLM:\SOFTWARE\Microsoft\Virtual Machine\External.

Those sent by the Child VM are in HKLM:\SOFTWARE\Microsoft\Virtual Machine\auto.

If the VM isn't running its Key/Value Pair Exchange Service does NOT persist the values. So stopped VMs won't return anything !.

### See Also:

Get-VMKVP

Remove-VMKVP

[MsVM\_KvpExchangeComponent on MSDN](http://msdn.microsoft.com/en-us/library/cc136848(VS.85).aspx)

## Remove-VMKVP

Removes a Key/Value pair from the set sent to one or more Virtual Machines.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to modify. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Key** *Required* |  | The Name of the key. |
| **Server** *Required* |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Removes a Key/Value pair from the set sent to one or more Virtual Machines.

### See Also:

Get-VMKVP

Add-VMKVP

[MsVM\_KvpExchangeComponent on MSDN](http://msdn.microsoft.com/en-us/library/cc136848(VS.85).aspx)

## Get-VMRemoteFXController

Gets the RemoteFx graphics controller(s) for one or more VM(s).

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** | (Input from pipeline) Wildcards | The Virtual Machine(s) to check This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified all VMs are checked. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |

### Description

Gets the RemoteFx graphics controller(s) for one or more VM(s).

### Examples

Get-VMRemoteFXController "Tenby"

Returns a WMI object representing the RemoteFX Controller on VM named "Tenby" (if there is one).

### See Also:

Add-VMRemoteFXController

Remove-VMRemoteFXController

## Set-VMRemoteFXController

Modifies or adds a RemoteFX graphics controller on a VM.

### Description

If a VM already has a RemoteFX graphics controller, Set-VMRemoteFXController modifies its settings. If not, Set-VMRemoteFXController calls Add-VMRemoteFXController to install one.

### Examples

Set-VMRemoteFXController -VM Tenby -monitors 2

Sets the RemoteFx controller on the VM named Tenby to support 2 monitors. If there is no controller one will be created using the default resolution.

Set-VMRemoteFXController -RFXRASD (Get-VMRemoteFXController) -Resolution 3

Gets all pre-existing RemoteFX controllers on the local server and resets their resolution to 1600x1200.

### See Also:

Get-VMRemoteFXController

Remove-VMRemoteFXController

Add-VMRemoteFXController

## Add-VMRemoteFXController

Adds a new RemoteFX graphics controller to a VM.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to which graphics cards should be added. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |
| **Resolution** |  | Remote FX supports 4 resolutions listed below. The Default is 1024x768 The RemoteFxResoultion ENUM allows either the text below or the associated number to be specified Display1024x768 = 0, Display1280x1024 = 1, Display1600x1200 = 2, Display1920x1200 = 3. |
| **Monitors** |  | Remote FX supports clients with 1 to 4 monitors. The default is 1. |

### Description

Server 2008 R2-SP1 introduced accelerated graphics controllers. These need to be added explicitly to VMs. This command adds the RemoteFX controller to a VM. A VM may only have a single controller and attempting to add a second one will cause an error. Set-VMRemoteFXController is safer to use because it will add or modify as appropriate.

### Examples

Add-VMRemoteFXController -VM Tenby -monitors 2 -resolution Display1920x1200

Adds a RemoteFx controller supporting 2 monitors with 1920x1200 resolution to the VM named tenby.

Get-VM "Win7\*" -stopped | Add-VMRemoteFXController -resolution[RemoteFxResoultion]::Display1920x1200

Adds a RemoteFx controller supporting a single monitor with 1920x1200 to Stopped VMs named "WIN7" The controller will support a single monitor and the resolution is set using the [RemoteFxResoultion] enum.

### See Also:

Get-VMRemoteFXController

Remove-VMRemoteFXController

Set-VMRemoteFXController

## Remove-VMRemoteFXController

Removes the RemoteFX display controller(s) from one or more VMs.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) from which the graphics card(s) should be removed. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |

### Description

Removes the RemoteFX display controller(s) from one or more VMs. If no VM is present on the specified VM, the command completes without error.

### Examples

Remove-VMRemoteFXController "Tenby"

Removes the RemoteFX Controller from the VM named "Tenby".

Get-VM -server Node1,Node2 -Stopped | Remove-VMRemoteFXController -Force

Gets all VMs in a Stopped state on the cluster nodes "node1" and "Node2" and removes any RemoteFxControllers installed on them.

### See Also:

Add-VMRemoteFXController

Get-VMRemoteFXController

## Get-VMNic

Returns information about Network Interface Cards.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to query. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified the information will be returned for all VMs on the server. If no VM is specified, NICs from all VMs Will be returned. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer is used. An array of server names by be passed but wildcards cannot be used. |
| **Legacy** |  | Select legacy (emulated) NICs. If the user specifies both -legacy and -VMbus OR neither then all types of NIC Will be returned. |
| **VMBUs** |  | Select VM-Bus(synthetic) NICs. If the user specifies both -legacy and -VMbus OR neither then all types of NIC Will be returned. |

### Description

Returns information about legacy and/or VMbus Network Interface Cards on one or more Virtual Machine(s).

### Examples

Get-VMNic $core -legacy -VMbus

Returns both Legacy and VMbus NICs found on the VM pointed to by $core.

### See Also:

Select-VMNIC

Add-VMNic

Remove-VMNic

[MsVM\_EmulatedEthernetPortSettingData Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136836(VS.85).aspx)

[MsVM\_SyntheticEthernetPortSettingData Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136914(VS.85).aspx)

## Select-VMNic

Allows the user to select a Network Interface Cards connected to a VM.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to query. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified, NICs from all VMs Will be returned. |
| **Server** *Required* |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer is used. An array of server names by be passed but wildcards cannot be used. |
| **Multiple** |  | If specified allows the user to select multiple NICs. This is particularly useful when multiple VMs are specified. |

### Description

If a VM has only a single Network Interface Card, that NIC will be returned. If it has more than one, a list will be displayed to allow the user to make a selection.

### Examples

Select-VMnic $Core

Allows the user to choose a single NIC from the NICs on the VM pointed to by $core.

### See Also:

Get-VMNIC

Add-VMNic

Remove-VMNic

## Add-VMNic

Creates a new legacy or VM-bus Network Interface Card on a Virtual Machine.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to modify. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Virtualswitch** | Wildcards | The Virtual switch to connect. This may be a WMI object representing a switch or a string object containing the name of a switch (if a wildcard is used it must resolve to a single switch). If not specified, the new NIC is created in a disconnected state. |
| **MAC** |  | MAC address. If not specified the Network card will be assigned a MAC address automatically, the first time it is used. |
| **GUID** |  | VM-Bus NICS are identified by a GUID. If a new VM is created, using an existing VHD file the OS in the VM will see a new NIC unless the original GUID is used. |
| **Server** *Required* |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer is used. An array of server names by be passed but wildcards cannot be used. |
| **Legacy** |  | Create a legacy (emulated) NIC. If not specified a VM-bus (Synthetic) NIC Will be created. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Creates a new Network Interface Card on a virtual machine. It can create Legacy (emulated) or VM-bus (Synthetic) NICs, assign a fixed MAC address, and specify the GUID which identifies a VM-bus NIC.

### Examples

Add-VMNic $core (select-VMSwitch)

Adds a VM-bus nic to the VM pointed to by $core , choosing the connection from a list of switches.

Add-VMNIC "tenby" (Select-VMswitch) -legacy

Adds a legacy nic to the VM named Tenby on the local server, choosing the connection from a list of switches.

get-VM core-% -Server James-2008 | add-VMnic -virtualSwitch "Internal Virtual Network" -legacy

Adds a legacy nic to those VMs on Server James-2008 which have names beginning Core- and binds them to the "Internal virtual network" switch.

### See Also:

Get-VMNIC

Remove-VMNic

[MsVM\_EmulatedEthernetPortSettingData Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136836(VS.85).aspx)

[MsVM\_SyntheticEthernetPortSettingData Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136914(VS.85).aspx)

## Remove-VMNic

Removes a Network Interface Card from a VM.

### Parameters

|  |  |  |
| --- | --- | --- |
| **NIC** *Required* | (Input from pipeline) | The NIC to remove from the VM. |
| **VM** |  | The VM does not need to be passed to the function (it is determined from the NIC) but is retained for compatibility reasons. |
| **Server** |  | The Server does not need to be passed to the function (it is determined from the NIC) but is retained for compatibility reasons. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Removes a Network Interface Card from a VM.

### Examples

Remove-VMNIC $core (Select-VMNic $core)

Removes a NIC on server pointed to by $core, if there are multiple NICs the user will prompted to select one.

Get-VMNic -legacy -VM "Lab\*" | remove-VMNic -force

Gets all the legacy NICS on virtual machines with names beginning "Lab" and removes them without prompting.

### See Also:

Add-VMNIC

Get-VMNIC

Select-VMNIC

[MsVM\_EmulatedEthernetPortSettingData Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136836(VS.85).aspx)

[MsVM\_SyntheticEthernetPortSettingData Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136914(VS.85).aspx)

## Set-VMNICAddress

Changes the MAC address of a Network Interface Card.

### Parameters

|  |  |  |
| --- | --- | --- |
| **NIC** *Required* | (Input from pipeline) | An object representing the NIC to be modified. |
| **MAC** *Required* |  | MAC address. This must be 12 Hex digits. |
| **VM** |  | The VM does not need to be passed to the function (it is determined from the NIC) but is retained for compatibility reasons. |
| **Server** |  | The Server does not need to be passed to the function (it is determined from the NIC) but is retained for compatibility reasons. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc, with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Normally the MAC address of a new Network Interface Card is set the next time its VM boots. The address is assigned from a range held by the server which hosts the VM However the address can be set or changed manually.

### Examples

Set-VMNICAddress $core (Select-VMNic $core) "00155D010101"

Sets the MAC address of a NIC on the pointed to by $core, if there are multiple NICs the user will prompted to select one.

### See Also:

Get-VMNic

New-Nic

Select-VMNic

[MsVM\_EmulatedEthernetPortSettingData Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136836(VS.85).aspx)

[MsVM\_SyntheticEthernetPortSettingData Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136914(VS.85).aspx)

## Get-VMNicSwitch

Returns the switch connected to a a Network Interface Card.

### Parameters

|  |  |  |
| --- | --- | --- |
| **NIC** *Required* | (Input from pipeline) | An object representing the a Network Interface Card whose switch is sought. |

### Description

Returns the WMI object representing the Virtual Network Switch connected to a Network Interface Card.

### Examples

(Get-VMNic $VM -legacy -VMbus | get-VMNicSwitch) | foreach-object {$\_.elementName}

Returns names of Switches used by the VM pointed to by $core.

### See Also:

Get-VMNIC

Get-VMSwitch

## Set-VMNICSwitch

Connects a Network Interface Card to a Virtual switch.

### Parameters

|  |  |  |
| --- | --- | --- |
| **NIC** *Required* |  | The NIC to Modify. |
| **Switch** *Required* |  | The Virtual switch to connect. This may be a WMI object representing a switch or a string object containing the name of a switch (if a wildcard is used it must resolve to a single switch). |
| **VM** *Required* | Wildcards | The VM does not need to be passed to the function (it is determined from the NIC) but is retained for compatibility reasons. |
| **VirtualSwitch** |  | The Server does not need to be passed to the function (it is determined from the NIC) but is retained for compatibility reasons. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc, with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Connects a Virtual Network Interface Card to a different a virtual switch. If no switch is specified, the NIC is disconnected.

### Examples

Set-VMNICSwitch $core (Select-VMNic $core) (Select-VMswitch $core.\_\_server)

Re-connects a NIC on the VM pointed to by $core, If there are multiple NICs the user will prompted to select one, and they will be prompted to select a switch if there is more than one.

### See Also:

Get-VMNicSwitch

SelectVMSwitch

## Get-VMNicVLAN

Gets the VLAN ID associated with a Network Interface Card.

### Parameters

|  |  |  |
| --- | --- | --- |
| **NIC** *Required* | (Input from pipeline) | An object representing the NIC to query - may be passed via the pipeline. |

### Description

Gets the VLAN ID associated with a Network Interface Card.

### Examples

get-VMnic | ForEach-Object{Add-Member -input $\_ -MemberType noteproperty -Value $(Get-VMNICVLAN $\_) -Name "VLAN" -PassThru}

Gets all available NICs and adds the VLAN as a property of each.

### See Also:

Set-VMNicVLAN

[MsVM\_BindsTo Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136819(VS.85).aspx)

[MsVM\_NetworkElementSettingData Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136858(VS.85).aspx)

## Set-VMNicVLAN

Sets the VLAN ID associated with a Network Interface Card.

### Parameters

|  |  |  |
| --- | --- | --- |
| **NIC** *Required* | (Input from pipeline) | An object representing the NIC to update, may be passed via the pipeline. |
| **VLANID** |  | The ID to Assign to the VLAN. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Sets the VLAN ID associated with a Network Interface Card.

### See Also:

Get-VMNicVLAN

[MsVM\_BindsTo Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136819(VS.85).aspx)

[MsVM\_NetworkElementSettingData Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136858(VS.85).aspx)

[MSVM\_hostedAccessPoint class on MSDN](http://msdn.microsoft.com/en-us/library/cc542581(VS.85).aspx)

## Add-VMSCSIController

Adds a Synthetic SCSI controller to one or more VM(s).

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s). This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Name** |  | The display name for the controller. The Default name is held in the preference variable lstr\_VMBusSCSILabel and can be changed. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Adds a Synthetic SCSI controller to one or more VM(s).

### Examples

Add-VMSCSIController $tenby

Adds a VMBus SCSI Controller to VM whose info is in $tenby.

Get-VM Core-% -server james-2008 | Add-VMSCSIController

Adds a SCSI Controller to all VMs whose names begin CORE- on the server named "James-2008".

## Remove-VMSCSIcontroller

Removes a SCSI controller from one or more virtual Machine(s).

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to modify . This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **ControllerID** |  | The Zero-based ID of the intended disk controller. If not specified 0 is used. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none"). |

### Description

Removes a SCSI controller from one or more virtual Machine(s).

### Examples

Remove-VMSCSIController $tenby 0

Remove the first VMBus SCSI Controller to VM whose info is in $tenby.

### See Also:

Get-VMDiskcontroller

Add-VMSCSIController

## Get-VMDiskController

Returns Disk controllers attached to a VM.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to query . This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If not specified al VMs are queried. |
| **ControllerID** |  | The Zero-based ID of the intended disk controller. If not specified, all controllers of the selected type are returned. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **SCSI** |  | If specified indicates that only SCSI devices is intended. If neither IDE nor SCSI is specified (or both) then both types of controller are returned. |
| **IDE** |  | If specified indicates that only IDE devices is intended. If neither IDE nor SCSI is specified (or both) then both types of controller are returned. |

### Description

Returns MSVM\_ResourceAllocationSettingData objects for the disk controller(s). The accompanying format.ps1xml file formats MSVM\_ResourceAllocationSettingData objects to show the VM’s element name, the resource’s Element name, its subtype (Microsoft Emulated IDE Controller or Microsoft Synthetic SCSI controller) and its connection (empty for disk controllers).

### Examples

Get-VM -server James-2008| Get-VMDiskController -IDE -SCSI

Returns all the DiskControllers for all the VMs on Server James-2008.

Get-VMDiskController $Tenby -SCSI -controllerID 0

Returns SCSI controller 0 in the VM pointed to by $Tenby.

### Notes

To return the nth SCSI and nth IDE controller BOTH switches must be specified as well as the controller ID.

### See Also:

Get-VMDriveByController

Add-VMScsicontroller

## Get-VMDriveByController

Returns the drives attached to a given disk controller.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Controller** *Required* | (Input from pipeline) | The Controller parameter is either a single MSVM\_ResourceAllocationSettingData WMI object which represents an IDE or SCSI controller or an array which contains these. |
| **Lun** | Wildcards | The Zero-based Logical Unit number which identifies a drive on a disk controller. If no LUN is passed, all drives are returned. |

### Description

Returns the drives attached to a given disk controller.

### Examples

Get-VMDiskController "Tenby" -server "James-2008" -IDE -controllerID 0 | Get-VMDriveByController

Gets IDE disk controller 0 for the VM named "Tenby" on the server named "james-2008" and then gets the Drives attached to it.

$drive=Get-VMDriveByController $controller –lun 0

Gets the first disk attached to the Controller specified by $controller and stores the result in $drive.

### See Also:

Get-VMDiskController

Get-VMDiskByDrive

## Add-VMDrive

Adds a drive to a controller on the specified VM.

### Description

Adds a drive (either a DVD or Hard disk drive) to the specified LUN slot on the specified IDE controller of a VM.

### Examples

Add-VMDRIVE "tenby" 1 1 -server james-2008

Adds a virtual DVD to IDE controller 1, disk slot 1 on the VM named Tenby on Server James-2008.

Add-VMDRIVE $tenby 0 3 -SCSI

Adds a Virtual Hard Disk drive to SCSI controller 0, LUN 3 on the VM whose info is in $tenby.

Get-VM Core-% | Add-VMDRIVE -controllerID 0 -lun 1 -DVD

Adds a DVD drive to IDE controller 0, disk slot 1 on all the VMs on the local server whose name begins with CORE-.

## Remove-VMdrive

Removes a drive and/or the disk image in it from a VM.

### Description

Removes a drive and/or the disk image in it from a VM.

### Examples

Remove-VMdrive "Tenby" 0 1 -SCSI -DiskOnly -Server "James-2008"

Removes the disk mounted in Scsi Controller 0, Lun 1 on the VM named tenby on the server named james-2008.

Remove-VMdrive $Core 1 1 -IDE

Removes the disk drive and any associated disk from IDE controller 1, Lun 1 on the VM point to by $core.

Remove-VMdrive "Tenby" 0 1 -SCSI -DiskOnly -Server "James-2008"

Remove the Disk from the drive at device 1 of SCSI controller 0 of the VM named "Tenby" on the Server "James-2008".

Remove-VMdrive $Core 1 1 -IDE

Remove the Disk and drive at device 1 of IDE controller 1 in the VM pointed to by $core.

get-VMdisk | where {$\_.diskpath -eq "C:\Windows\system32\VMguest.iso"} | foreach {remove-VMDrive -diskonly -VM $\_.VMelementName -c $\_.controllerId -L $\_.driveLun}

Find all instances of the guest extensions ISO and remove them.

### See Also:

Add-VMDrive

Get-VMDisk

Get-VMDiskController

Get-VMDriveByController

## Get-VMDiskByDrive

Returns the disk mounted in a given drive.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Drive** *Required* | (Input from pipeline) | A single MSVM\_ResourceAllocationSettingData WMI object which represents a hard-disk drive or DVD drive or an array which contains these. If an array is passed the function calls itself recursively for each item. |

### Description

Returns the disk mounted in a given drive.

### Examples

Get-VMDiskController "Tenby" -server "James-2008" -IDE -controllerID 0 | Get-VMDriveByController | Get-VMdiskByDrive

Gets the disks in the drives attached to IDE controller 0 in the VM named Tenby on Server James-2008.

get-VMdiskByDrive $drive

Gets the disk in the drive pointed to by $drive.

### See Also:

Get-VMDriveByController

## Add-VMDisk

Adds a disk image to a VM, mounting it in a drive.

### Description

Mounts a disk (usually a VHD file, but also an ISO file or the pass-through path to an optical disk) into an drive on a VM. If the drive does not already exist, it will be created.

### Examples

Add-VMDisk $tenby 0 1 "C:\update.iso" -optical

Adds a DVD image C:\update.iso, to disk 1, controller 0 on the VM whose info is in $tenby.

Add-VMDisk $tenby 0 0 ((get-VHDdefaultPath) +"\tenby.VHD")

Adds a virtual hard disk named tenby.VHD in the Default folder , to disk 0, controller 0 on the VM whose info is in $tenby. If a name is passed with no path the Default VHD Path will be added so this command could be written more simply.

Add-VMDisk $tenby -Controller 0 -lun 0 "tenby.VHD"

A better version of example 2.

Get-VM "Core\*" | Add-VMDisk -path {$\_.ElementName +".VHD"}

Gets all the virtual machines with names being with "CORE". For each one a hard disk is added to IDE controller 0, LUN 0. The VHD files are in the Default folder, and share their name with the VM.

## Get-VMDisk

Returns all disks attached one or more VMs,.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to query . This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If not specified al VMs are queried. |
| **snapshot** |  | If the snapshot switch is included disks attached to snapshots are included. |

### Description

The function returns objects with 10 properties VMElementname The display name of the VM VMGUID Its GUID (ComputerSystem’s "name" property) ControllerName The Element name property of the disk controller object ControllerInstanceID The WMI Instance ID of the disk controller object ControllerID The logical controller number DriveName The element Name of the drive object DriveInstanceID The WMI InstanceID for the drive DriveLun The Slot occupied by the drive (disk's address property) of the disk object) DiskPath The path to the disk (disk's connection property ) DiskName The Element name property of the disk object DiskInstnaceID The WMI Instance ID for the disk.

### Examples

Get-VMDisk (Select-VM -server "James-2008" -multi) | format-table -autosize -property VMname, DriveName, @{Label="Conected to"; expression={"{0,5} {1}:{2}" –f $\_.Controllername.split(" ")[0], $\_.ControllerID,$\_.DriveLun }} , DiskPath

Gets the disks (without snapshot disks) for the chosen VMs on Server James-2008, and returns the result as a table with the VMName, Drive name, the connection path, and the DiskPath.

Get-VMDisk | foreach {$\_.diskpath}

Returns a list of disks in use.

Get-VMDisk | where {$\_.ControllerName -match "^IDE"}

Returns a list of disks attached to IDE controllers.

### See Also:

Get-VMDiskController

Get-VMDriveByController

Get-VMDiskByDrive

Get-VHDInfo

## Set-VMDisk

Changes the Disk mounted in a Drive on a VM.

### Description

Changes the Disk mounted in a Drive on a VM.

### Examples

Set-VMDisk Tenby 0 1 (Get-WmiObject -Query "Select \* From win32\_cdromdrive Where ID='D:' " ).deviceID

Sets the DVD on controller 0, device 1 for the VM named "Tenby" on the local Server to point to physical drive D: on the host.

Set-VMDisk $Core 0 0 "\\?\Volume{d1f72a03-d43a-11dc-8bf1-806e6f6e6963}\Virtual Hard Disks\Core.VHD"

Sets the Disk on controller 0, device 0 of the VM pointed to by $core to Core.VHD using GUID (not drive letter) notation.

## Add-VMNewHardDisk

Creates and attaches a new Virtual Hard Disk in one command.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s). This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **ControllerID** |  | The Zero-based ID of the intended disk controller. If not specified 0 is used. |
| **LUN** |  | The Zero-based Logical Unit number which identifies a drive on a disk controller. If not Specified 0 is used. |
| **VHDPath** |  | The path to the VHD file. If not specified the name of the VM will be used. If a script block is passed it will be evaluated to determine the path. If the .VHD file extension is omitted it will be added and if only a file name is passed the server’s default VHD folder (not the current working directory) will be assumed. |
| **Size** |  | The Size of the disk in bytes. If not specified 127GB will be used. |
| **ParentVHD** |  | To create a differencing disk the parent-disk path must be specified and the size and fixed parameters are then ignored. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Fixed** |  | Indicates a fixed disk is required. If neither ParentVHD nor fixed are specified a dynamic VHD will be created. |
| **SCSI** |  | If specified indicates that a SCSI device is intended. If not specified IDE is used. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Add-VMNewHardDisk combines New-VHD, Add-VMDrive and Add-VMDisk. It calls NEW-VHD with ParentDisk, Size, Fixed, VHDPath and Server parameters. If no VHDPath is passed the VHD is created in the default folder using a name which matches the VM. It then calls Add-VMDrive with the Server, VM, SCSI, Controller and LUN parameters to create the drive Finally it calls Add-VMDisk with VM, controllerID, Lun, VHDPath, Server and SCSI parameters. Note that the function assumes that the controller exists.

### Examples

Add-VMNewHardDisk -VM $VM -controllerID 0 -lun 3 -VHDpath "$(get-VHDdefaultPath)\foo31.VHD" -size 20gb -scsi

Adds a 20GB dynamic disk, named foo31.VHD in the default folder, to the VM defined in $VM, on SCSI controller 0, Lun 3.

Get-VM "Core\*" | Add-VMNewHardDisk -lun 0 -VHDpath {$\_.elementName + -"os"} -size 20gb -Fixed Get-VM "Core\*" | Add-VMNewHardDisk -lun 1 -VHDpath {$\_.elementName + "-Data"}

Gets all the machines with names that begin with "Core"; Adds two disk to each one a 20GB fixed disk on Lun 0 and a 127GB dynamic disk on Lun 1. The disks will share the name of the VM , suffixed with "-OS" and "-Data" respectively.

## Add-VMPassThrough

Connects a Passthrough disk to a VM.

### Description

Hyper-V allows disks which are seen as Off-line by the parent partition to be connected to a LUN in VM, bypassing the use of VHD files. Add-VMPassthrough makes the connection.

### Examples

Add-VMPassThrough $VM 0 1 (Select-VMPhysicaldisk)

If there are no physical disks available to Hyper-v , (i.e. connected and marked offline), this will do nothing. If there is exactly one disk that is used. If there is more than one the user is prompted to select one. The physical disk is bound to IDE controller 0 , Lun 1 on the VM held in $VM.

## Select-VMPhysicalDisk

Allows the user to make a choice between disks to become a pass-through disk.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Server** |  | Specifies the Hyper-V server on which a the physical disk will be sought. By default "." , the local computer, is used. Only a single server may be specified. |

### Description

Allows the user to make a choice between disks to become a pass-through disk. If more than one disk is available a menu will be displayed.

### Examples

Add-VMPassThrough $VM 0 1 (Select-VMPhysicaldisk)

If there are no physical disks available to Hyper-v (i.e. connected and marked offline), this will do nothing. if there is exactly one disk that is used. If there is more than one the user is prompted to select one The physical disk is bound to IDE controller 0 , Lun 1 on the VM held in $VM.

### See Also:

Add-VMPassthroughDisk

[MsVM\_ResourcePool Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136879(VS.85).aspx)

## Get-VMFloppyDisk

Returns the floppy disk(S) attached to one or more VMs.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to query . This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If not specified al VMs are queried. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |

### Description

Returns MSVM\_ResourceAllocationSettingData objects for the Floppy disk(s). The accompanying format.ps1xml file formats MSVM\_ResourceAllocationSettingData objects to show the VM’s element name, the resource’s Element name, its subtype (Microsoft virtual Floppy disk) and its connection (the path to the VFD file).

### Examples

Get-VMFloppyDisk (get-VM -server james-2008) | foreach {$\_.connection}

Produces a list of all the VFD files in the floppy drives of the VMs on the server James-2008.

### See Also:

Add-VMFloppyDisk

Remove-VMFloppyDisk

## Add-VMFloppyDisk

Adds a floppy disk to a VM.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s). This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Path** *Required* |  | The path to the VFD file to mount in the Floppy drive. This may be a string which holds the path to the VFD file, a script block which is evaluated to give the path of the VFD file. or a System.Io.FileInfo object representing the file. If the .VFD extension is missing from the path it will be added. If the path is a file name, the server's default VHD folder (not the current directory) will be used. Has an Alias of VFDpath. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

HyperV VMs are created a single floppy drive. Add-VMFloppyDisk mounts a virtual Floppy disk (VFD) image file in the drive.

### Examples

add-VMFloppyDisk $core "C:\Users\Public\Documents\Microsoft Hyper-V\Blank Floppy Disk\blank.VFD"

Adds a floppy disk to the machine Pointed to by $Core , the VFD being Blank.vfd.

Get-VM "Core\*" | add-VMFloppyDisk -path {$\_.elementName}

Gets all VMs with names which start "core" and adds a floppy disk to them. The VFD files be located in the default and will have the same names as the VMs.

## Remove-VMFloppyDisk

Removes floppy(s) disk attached to one or more VM(s).

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** *Required* | (Input from pipeline) Wildcards | The Virtual Machine(s) to modify . This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none"). |

### Description

Removes any floppy disk attached to the VM(s) provided. If no Floppy is present no error occurs.

### See Also:

Get-VMFloppyDisk

Add-VMFloppyDisk

## New-Vfd

Creates one or more virtual Floppy disk file(s).

### Parameters

|  |  |  |
| --- | --- | --- |
| **VFDPaths** *Required* | (Input from pipeline) | The path(s) to the VFD file(s) must be specified If the .VFD file extension is omitted it will be added and if only a file name is passed the server’s default VHD folder will be assumed (Hyper-V does not have the concept of a default VFD folder). |
| **Server** |  | Specifies the Hyper-V server on which the VFD(S) will be created By default "." , the local computer, is used. |
| **Wait** |  | Indicates that the command should wait until the disk creation job completes. |

### Description

Creates one or more virtual Floppy disk file(s).

### Examples

New-VFD "Floppy.VFD"

Creates a new floppy disk named FLOPPY.VFD in the default folder.

### See Also:

Add-VMFlopyDisk

[CreateVirtualFloppyDisk Method of the MsVM\_ImageManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136784(VS.85).aspx)

## New-VHD

Creates one or more Virtual Hard Disk file(s).

### Parameters

|  |  |  |
| --- | --- | --- |
| **VHDPaths** *Required* | (Input from pipeline) | The path(s) to the VHD file(s) must be specified. If the .VHD file extension is omitted it will be added and if only a file name is passed the server’s default folder (not the current working directory) will be assumed. |
| **Size** |  | The Size of the disk in bytes. If not specified 127GB is used, as in the GUI. |
| **ParentVHD** |  | To create a differencing disk the parent disk's path must be specified, in which case and the size and fixed parameters are ignored. This parameter has aliases of ParentDiskPath and ParentPath, and can be a string or an object which can be converted to a string; if a scriptblock is specified it will be evaluated. |
| **Server** |  | Specifies the Hyper-V server on which the VHD(s) will be created By default "." , the local computer, is used. Only a single server may be specified. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Fixed** |  | Indicates a fixed disk is required. If neither ParentVHD nor fixed is specified a dynamic VHD will be created. |
| **Wait** |  | Indicates that the command should wait until the disk creation job completes. This can take several minutes in the case of large fixed disks. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Creates one or more Virtual Hard Disk file(s).

### See Also:

Add-VMDisk

Get-VMDisk

Add-VMNewHardDisk

[MsVM\_ImageManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136845(VS.85).aspx)

## Get-VHDDefaultPath

Gets the default path for Virtual Hard Disk (VHD) files.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Server** |  | Specifies the Hyper-V server to query By default "." , the local computer, is used. Only a single server may be queried. |

### Description

The default path for VHD files is "C:\users\public\documents\hyper-v\Virtual Hard Disks" Get-VHDDefaultPath obtains the path for a given server.

### See Also:

New-VHD

Get-VHD

[MsVM\_VirtualSystemManagementServiceSettingData Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136941(VS.85).aspx)

## Get-VHD

Gets VHD files from a specified folder on a hyper-v server.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Server** |  | Specifies the Hyper-V server to query By default "." , the local computer, is used. Only a single server may be queried. |
| **Paths** |  | The folder(s) to search for VHD files on the specified server(s). If not specified the default VHD folder will be queried. |

### Description

Gets VHD files from a specified folder on a hyper-v server (by default the local server, but a remote server may be specified).  
 The default path for VHD files is "C:\users\public\documents\hyper-v\Virtual Hard Disks"   
Get-VHDDefaultPath obtains the path for a given server.

### Notes

Other -VHD commands can accept the output of Get-VHD as piped input.

In this situation the server parameter MUST be specified in the next function.

## Get-VHDInfo

Gets detailed information about one or more VHD files.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VHDPaths** *Required* | (Input from pipeline) | A string which contains the name of a VHD file or an array which contains these. If no VHDPath is passed as a parameter, the function looks for input to be piped to it. If an object is piped, the command will look for Fullname, Path or Diskpath properties, in that order, to use as the path. These names may also be used as aliases for -Path. If the .VHD file extension is omitted it will be added and if the path only contains a file name then the server’s default VHD folder, (not the current working directory) will be assumed. When the command is run against the local Server, wildcards and relative paths will be expanded. |
| **Server** |  | Specifies the Hyper-V server on which the VHD file(s) reside(s). By default "." , the local computer, is used. Only a single server may be queried. |

### Description

For each VHD requested, Get-VHDInfo returns an object with the following properties VHDPath The path to the file, Path An alternate name for VHDPath, FileSize The size of the file on disk InSavedState Indicates whether the disk is associated with a VM in a saved state InUse Indicates whether the disk is mounted MaxInternalSize The size as seen by the virtual machine ParentPath The parent for a differencing disk (empty for other disks) Type 2="Fixed",3="Dynamic",4="Differencing" TypeName The type Number converted to text.

### Examples

cd (Get-VHDDefaultPath) ; dir \*.VHD | get-VHDinfo

Moves to the default folder for VHDs gets all the VHD files and passes them into Get-VHDInfo.

(Get-VHDInfo 'C:\Users\Public\Documents\Microsoft Hyper-V\Virtual Hard Disks\Core.VHD').parentPath

Returns the parent path of a single differencing disk.

Get-VMDisk "core%" | forEach {Get-VHDInfo $\_.Diskpath} | measure-object -Sum filesize

Gets all the disks on virtual machines with names beginning with CORE, gets the disk info for each one, and calculates the sum their file sizes.

### See Also:

Get-VHD

Get-VMDisk

[GetVirtualHardDiskInfo Method of the MsVM\_ImageManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136797(VS.85).aspx)

## Test-VHD

Tests the working state of a disk - for example if its parent can be found.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VHDPaths** *Required* | (Input from pipeline) | A string which contains the name of a VHD file or an array which contains these. If no VHDPath is passed as a parameter, the function looks for input to be piped to it. If an object is piped, the command will look for Fullname, Path or Diskpath properties, in that order, to use as the path. These names may also be used as aliases for -Path. If the .VHD file extension is omitted it will be added and if the path only contains a file name then the server’s default VHD folder, (not the current working directory) will be assumed. When the command is run against the local Server, wildcards and relative paths will be expanded. |
| **Server** |  | Specifies the Hyper-V server on which the VHD file resides. By default "." , the local computer, is used. Only a single server may be specified. |

### Description

Tests the working state of a disk - for example if its parent can be found.

### Examples

dir "$(Get-VHDDefaultPath)\\*.VHD" | Test-VHD

Gets all the VHD files in the default folder and checks them.

Get-VMDisk | %{$\_.DiskPath} | where {$\_.endswith(".VHD")} | Test-VHD

Gets all disk on the VMs and validates them.

## Connect-VHDParent

Reattaches a differencing VHD to its parent.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VHDPaths** *Required* | (Input from pipeline) | A string which contains the name of a VHD file or an array which contains these. If no VHDPath is passed as a parameter, the function looks for input to be piped to it. If an object is piped, the command will look for Fullname, Path or Diskpath properties, in that order, to use as the path. These names may also be used as aliases for -Path. If the .VHD file extension is omitted it will be added and if the path only contains a file name then the server’s default VHD folder, (not the current working directory) will be assumed. When the command is run against the local Server, wildcards and relative paths will be expanded. |
| **ParentPath** *Required* |  | The location of the moved or recovered parent VHD file. This can be a string object or a script block which is evaluated to give the name. It has aliases ParentPath and ParentDiskPath. |
| **Server** |  | Specifies the Hyper-V server on which the VHD resides. By default "." , the local computer, is used. |
| **Wait** |  | If the wait switch is specified the function will display a progress indicator until the job completes. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

If the parent of differencing disk is moved or renamed, the VHD must be reattached to it before it can be used. The parent disk must be an identical copy or the same file as the original.

### See Also:

Get-VHD

Get-VHDInfo

Test-VHD

[ReconnectParentVirtualHardDisk Method of the MsVM\_ImageManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136960(VS.85).aspx)

## Compress-VHD

Compacts one or more dynamic VHD files.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VHDPaths** *Required* | (Input from pipeline) | A string which contains the name of a VHD file or an array which contains these. If no VHDPath is passed as a parameter, the function looks for input to be piped to it. If an object is piped, the command will look for Fullname, Path or Diskpath properties, in that order, to use as the path. These names may also be used as aliases for -Path. If the .VHD file extension is omitted it will be added and if the path only contains a file name then the server’s default VHD folder, (not the current working directory) will be assumed. On the when the command is used against the local Server, wildcards and relative paths may be used. |
| **Server** |  | Specifies the Hyper-V server on which the VHD resides. By default "." , the local computer, is used. |
| **Wait** |  | If the wait switch is specified the function will display a progress indicator until the job completes. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Unused space in dynamic VHD files can be reclaimed. This does not affect the maximum size or the partitions and data inside the VHD.

### Examples

Compress-VHD (get-VHDdefaultPath) +"\tenby.VHD"

Compacts the VHD as a background job. You can check status with Get-WmiObject -NameSpace root\virtualization msVM\_storagejob | ft jobStatus, description, percentcomplete -auto.

### See Also:

Expand-VHD

Get-VHD

Merge-VHD

Convert-VHD

[CompactVirtualHardDisk Method of the MsVM\_ImageManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136773(VS.85).aspx)

## Convert-VHD

Creates a new VHD of a different type based on an existing VHD.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VHDPaths** *Required* | (Input from pipeline) | A string which contains the name of a VHD file or an array which contains these. If no VHDPath is passed as a parameter, the function looks for input to be piped to it. If an object is piped, the command will look for Fullname, Path or Diskpath properties, in that order, to use as the path. These names may also be used as aliases for -Path. If the .VHD file extension is omitted it will be added and if the path only contains a file name then the server’s default VHD folder, (not the current working directory) will be assumed. When the command is run against the local Server, wildcards and relative paths will be expanded. |
| **DestPath** *Required* |  | The location where the resulting file should be saved. This can be a string containing the path or a scriptblock which is evaluated to provide the path. If the .VHD file extension is omitted it will be added and if the path only contains a file name then the server’s default VHD folder, (not the current working directory) will be assumed. |
| **Type** *Required* |  | An enum indicating the type of VHD. If a valid integer (2,3) or a valid string "Fixed","Dynamic" is passed, it will be converted to an enum. |
| **Server** |  | Specifies the Hyper-V server on which the VHD resides. By default "." , the local computer, is used. |
| **Wait** |  | If the wait switch is specified the function will display a progress indicator until the job completes. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Creates a new VHD of a different type based on an existing VHD; for example takes a differencing disk and creates a new fixed or dynamic disk.

### Examples

convert-VHD core temp -type [VHDType]::dynamic

Will merge a differencing disk "CORE.VHD" in the default folder (and its parent(s) ) into a new disk called temp.VHD also in the default folder.

Convert-VHD "$( Get-VHDDefaultPath )\Temp.VHD" F:\backups\MyDisk.VHD -type [VHDType]::fixed

Will convert a disk to a fixed one on a different drive.

dir \*.VHD | get-VHDinfo | where-object {$\_.type -eq 3} | convert-VHD -type "Fixed" -path {$\_.replace(".VHD","-FIXED.VHD")}

Finds VHD files in the current folder and isolates the Dynamic ones, and converts them to fixed files with -FIXED appended to the file name.

pushd (Get-VHDDefaultPath) ; dir \*.VHD | get-VHDinfo | where-object {$\_.type -eq 3} | foreach {convert-VHD $($\_.path) '.\temp.VHD' -type 2 -wait ; del $($\_.path) ; ren temp.VHD $($\_.path)} ; popd

Move to the default VHD folder, get the VHD files, isolate the dynamic ones, convert them to a fixed size one named temp , delete the original , rename temp to the original name.

### Notes

Note that moving a disk type from differencing to anything else is a convert not a merge.

### See Also:

New-VHD

Merge-VHD

[ConvertVirtualHardDisk Method of the MsVM\_ImageManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136775(VS.85).aspx)

## Expand-VHD

Increases the size of a VHD.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VHDPaths** *Required* | (Input from pipeline) | A string which contains the name of a VHD file or an array which contains these. If no VHDPath is passed as a parameter, the function looks for input to be piped to it. If an object is piped, the command will look for Fullname, Path or Diskpath properties, in that order, to use as the path. These names may also be used as aliases for -Path. If the .VHD file extension is omitted it will be added and if the path only contains a file name then the server’s default VHD folder, (not the current working directory) will be assumed. When the command is run against the local Server, wildcards and relative paths will be expanded. |
| **Size** *Required* |  | The size for the disk in bytes. It must be bigger than the current size. The Hyper-V GUI only allows whole GB to be selected, but a fraction may be used. |
| **Server** |  | Specifies the Hyper-V server on which the VHD resides. By default "." , the local computer, is used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Wait** |  | If the wait switch is specified the function will display a progress indicator until the job completes. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

If a valid file is provided the function attempts to extend the VHD specified to the value specified in the size parameter, using the Image Management Service.

### Examples

Expand-VHD 'C:\users\Public\Documents\Microsoft Hyper-V\Virtual Hard Disks\Tenby.VHD' 22gb

Expands the named disk to 22GB in size.

### Notes

This will not expand the partition(s) on the disk, that needs to be done separately.

### See Also:

New-VHD

Get-VHDInfo

[ExpandVirtualHardDisk Method of the MsVM\_ImageManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136792(VS.85).aspx)

## Merge-VHD

Merges VHDs (from snapshots).

### Parameters

|  |  |  |
| --- | --- | --- |
| **VHDPaths** *Required* | (Input from pipeline) | A string which contains the name of a VHD file or an array which contains these. If no VHDPath is passed as a parameter, the function looks for input to be piped to it. If an object is piped, the command will look for Fullname, Path or Diskpath properties, in that order, to use as the path. These names may also be used as aliases for -Path. If the .VHD file extension is omitted it will be added and if the path only contains a file name then the server’s default VHD folder, (not the current working directory) will be assumed. When the command is run against the local Server, wildcards and relative paths will be expanded. |
| **DestPath** *Required* | (Input from pipeline) | The location where the resulting file should be saved. This can be a string containing the destination path or a script block which is evaluated to provide it. |
| **Server** |  | Specifies the Hyper-V server on which the VHD resides. By default "." , the local computer, is used. Only a single server may be specified. |
| **Wait** |  | If the wait switch is specified the function will display a progress indicator until the job completes. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

If a valid file is provided the function attempts to merge the differencing disk(s) specified by VHDpaths into the ancestor specified by DestPath.

### See Also:

New-VHD

Convert-VHD

[MergeVirtualHardDisk Method of the MsVM\_ImageManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136805(VS.85).aspx)

## Mount-VHD

Mounts a VHD file to make it appear to be a disk in the parent partition.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VHDPaths** *Required* | (Input from pipeline) | A string which contains the name of a VHD file or an array which contains these. If no VHDPath is passed as a parameter, the function looks for input to be piped to it. If an object is piped, the command will look for Fullname, Path or Diskpath properties, in that order, to use as the path. These names may also be used as aliases for -Path. If the .VHD file extension is omitted it will be added and if the path only contains a file name then the server’s default VHD folder, (not the current working directory) will be assumed. When the command is run against the local Server, wildcards and relative paths will be expanded. |
| **Partition** |  | If a partition number is provided it will be assigned a drive letter, according to the DriveLetter and No DriveLetter Parameters. |
| **DriveLetter** |  | If specified , and partition has been passed, the specified partition will be assigned the requested drive letter. Note an error will occur if a drive letter is specified and multiple VHDs are mounted. |
| **NoDriveLetter** |  | If specified the requested partition will be assigned no drive letter If a partition is specified with neither DriveLetter nor NoDriveLetter , it will be assigned a free drive letter by the system. |
| **Offline** |  | Specifies the disk is to be mounted offline. |

### Description

Mounts a VHD file to make it appear to be a disk in the parent partition.

### Examples

dir "$(Get-VHDDefaultPath)\\*.VHD" | Mount-VHD –offline

Gets all the VHD files in the default folder, and mounts them in an Offline state.

Mount-VHD tenby

Mounts tenby.VHD from the default folder, and brings the disk on line.

Mount-VHD –path "C:\users\public\documents\hyper-v\Virtual Hard Disks\tenby.VHD" –Partition 2 –letter H

Mounts tenby.VHD, brings the disk on line and assigns drive H: to the Second partition.

### See Also:

New-VHD

Dismount-VHD

[Mount Method of the MsVM\_ImageManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136811(VS.85).aspx)

## Dismount-VHD

Dismounts a previously mounted VHD.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VHDPaths** *Required* | (Input from pipeline) | A string which contains the name of a VHD file or an array which contains these. If no VHDPath is passed as a parameter, the function looks for input to be piped to it. If an object is piped, the command will look for Fullname, Path or Diskpath properties, in that order, to use as the path. These names may also be used as aliases for -Path. If the .VHD file extension is omitted it will be added and if the path only contains a file name then the server’s default VHD folder, (not the current working directory) will be assumed. When the command is run against the local Server, wildcards and relative paths will be expanded. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none"). |

### Description

If a valid file is provided then the VHD file at that location is dismounted using the Image Management Service.

### Examples

Dismount-VHD (get-VHDdefaultPath) +"\tenby.VHD"

Dismounts the VHD.

dir "$(Get-VHDDefaultPath)\\*.VHD" | Dismount-VHD

Attempts to Dismount all the disks in the folder - this will fail gracefully if they are not mounted.

### See Also:

Mount VHD

[Mount Method of the MsVM\_ImageManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136811(VS.85).aspx)

## Get-VHDMountPoint

Returns the mount point for a VHD file, if it is mounted.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VHDPaths** *Required* | (Input from pipeline) | A string which contains the name of a VHD file or an array which contains these. If no VHDPath is passed as a parameter, the function looks for input to be piped to it. If an object is piped, the command will look for Fullname, Path or Diskpath properties, in that order, to use as the path. These names may also be used as aliases for -Path. If the .VHD file extension is omitted it will be added and if the path only contains a file name then the server’s default VHD folder, (not the current working directory) will be assumed. When the command is run against the local Server, wildcards and relative paths will be expanded. |

### Description

Returns the mount point (drive letter) for a VHD file, if it is mounted.

### See Also:

Mount-VHD

[MsVM\_MountedStorageImage Class on MSDN](http://msdn.microsoft.com/en-us/library/cc542583(VS.85).aspx)

## Wait-ForDisk

Waits for disk to come on line.

### Parameters

|  |  |  |
| --- | --- | --- |
| **MountPoint** *Required* |  | The drive letter being sought. If the trailing colon is omitted, the command will add it. |
| **Attempts** |  | The number of times to look for the logical disk object before returning false. |
| **MSPause** |  | The number of milliseconds to pause between attempts. |

### Description

Waits for a predefined time to see if a Win32\_logicalDisk WMI object can be found with a given drive letter.

## Get-VMSwitch

Returns one or more Virtual Switch objects.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VirtualSwitchName** | (Input from pipeline) Wildcards | The name of a Switch being sought (which may include wildcards.) WMI queries use % as a wildcard, but the command will convert \* to %. Name can be used as an Alias for VirtualSwitchName (as well as contractions of it). |
| **Server** *Required* |  | Specifies the Hyper-V server(s) to search for the switch(es) By default "." , the local computer is used. An array of server names by be passed but wildcards cannot be used. |

### Description

If no name is specified returns all the Virtual Network Switches on a host, Otherwise returns the switch(es) which match the given name.

### See Also:

New-VMExternalSwitch

New-VMInternalSwitch

New-VMPrivateSwitch

Remove-VMSwitch

Select-VMSwitch

[MsVM\_VirtualSwitch Class on MSDN](http://msdn.microsoft.com/en-us/library/cc542584(VS.85).aspx)

## Select-VMSwitch

Returns a Virtual Switch objects.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Server** *Required* |  | Specifies the Hyper-V server on which the switches reside. By default "." , the local computer is used. An array of server names by be passed but wildcards cannot be used. |

### Description

If only one switch is found on a server, it is returned, if more than one exists a list is provided to allow the user to make a selection.

## Remove-VMSwitch

Deletes a virtual network switch.

### Parameters

|  |  |  |
| --- | --- | --- |
| **virtualSwitch** *Required* | (Input from pipeline) | A string containing the name for the switch to be deleted, or a WMI object representing the switch. It may be passed via the pipe. |
| **Server** |  | Specifies the Hyper-V server to query for virtual switches By default "." , the local computer is used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc, with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none"). |

### Description

Deletes a virtual network switch.

### See Also:

New-VMPrivateSwitch

New-VMExternalSwtich

New-VMInternalSwitch

[DeleteSwitch method of MsVM\_VirtualSwitchManagementService on MSDN](http://msdn.microsoft.com/en-us/library/cc136789(VS.85).aspx)

## New-VMPrivateSwitch

Creates a virtual network switch, which isn't bound to a NIC in the parent partition.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VirtualSwitchName** *Required* |  | The name for the new switch. |
| **Ports** |  | The number of ports to be created on the new switch. |
| **Server** |  | Specifies the Hyper-V server to query for network interfaces By default "." , the local computer is used. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Creates a virtual network switch, which isn't bound to a NIC in the parent partition.

### Examples

New-VMPrivateSwitch "VM network" -server "HVCore"

Creates a Switch on the server named "HVCore". The network will not be accessible in the host OS, and will be named "VM Network" in the Hyper-V administration tools.

### See Also:

New-VMInternalSwitch

New-VMExternalSwitch

Remove-VMSwitch

[CreateSwitch method of MsVM\_VirtualSwitchManagementService on MSDN](http://msdn.microsoft.com/en-us/library/cc136783(VS.85).aspx)

## New-VMInternalSwitch

Creates a virtual network switch, bound to a virtual NIC in the parent partition.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VirtualSwitchName** *Required* |  | The name for the new switch. |
| **Ports** |  | The number of ports to be created on the new switch. |
| **Server** |  | Specifies the Hyper-V server to query for network interfaces By default "." , the local computer is used. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Creates a virtual network switch, and a virtual NIC in the parent partition, and binds them together.

### Examples

New-VMInternalSwitch "Host and VM network"

Creates a Switch and virtual NIC in the host. The device name for the NIC in the host and the Network name in Hyper-V will be "Host and VM Network".

### See Also:

New-VMPrivateSwitch

New-VMExternalSwtich

Remove-VMSwitch

Remove-VMSwitchNIC

[CreateSwitch method of MsVM\_VirtualSwitchManagementService on MSDN](http://msdn.microsoft.com/en-us/library/cc136783(VS.85).aspx)

[CreateInternalEthernetPortDynamicMac method of MsVM\_VirtualSwitchManagementService on MSDN](http://msdn.microsoft.com/en-us/library/cc443602(VS.85).aspx)

## New-VMExternalSwitch

Creates a virtual network switch, bound to a physical network card.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VirtualSwitchName** *Required* |  | The name for the new switch. |
| **ExternalEthernet** *Required* |  | Either the name of a Network interface, passed as a string, or a WMI object representing an interface. It may be passed via the pipe. |
| **Ports** |  | The number of ports to be created on the new switch. |
| **Server** |  | Specifies the Hyper-V server where the switch should be created By default "." , the local computer is used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc, with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none"). |

### Description

Creates a virtual network switch, and binds it to a network card which is available in the parent partition.

### Examples

Select-VMExternalEthernet | New-VMExternalSwitch -VirtualSwitchName "Wired virtual Network"

Allows the user to choose if there are multiple available NICs and binds the selected one to a new switch. The device name for the NIC in created in the host and the Network name in Hyper-V will be "Wired virtual Network".

New-VMExternalSwitch -VirtualSwitchName "Wired virtual Network" -ext "Broadcom" -Server Core

Finds a Nic with a name beginning "Broadcom" on the server named "core", and binds it to a new switch. The device name for the NIC in created in the host and the Network name in Hyper-V will be "Wired virtual Network".

### See Also:

New-VMPrivateSwitch

New-VMExternalSwtich

Remove-VMSwitch

[SetupSwitch method of MsVM\_VirtualSwitchManagementService on MSDN](http://msdn.microsoft.com/en-us/library/cc136973(VS.85).aspx)

## Select-VMExternalEthernet

Selects an avaialable host network Interface.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Server** |  | Specifies the Hyper-V server to query for network interfaces By default "." , the local computer is used. |

### Description

When an external virtual network switch is created it needs to be told which network interface on the host computer it should be bound to. If there is more than one available, a list is presented to allow the user to make a selection.

### See Also:

NewVMExternalSwitch

[MsVM\_ExternalEthernetPort on MSDN](http://msdn.microsoft.com/en-us/library/cc136838(VS.85).aspx)

## Remove-VMSwitchNIC

Removes the parent partition NIC associated with a Virtual switch.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Name** *Required* |  | The name of the NIC to be removed. |
| **Server** |  | Specifies the Hyper-V server to query for the NIC . By default "." , the local computer is used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc, with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none"). |

### Description

Removes the parent partition Network Interface Card associated with a Virtual switch. A virtual NIC is always created in the parent partition for internal and external networks. In Hyper-V R2 there is an option to not allow a management partition to access the an external NIC used by VMs. The same result can be achieved by using remove-VMswitchNic.

### Notes

This is intended to be called from other commands, and not from the command line.

### See Also:

[DeleteInternalEthernetPort Method of the MsVM\_VirtualSwitchManagementService Class](http://msdn.microsoft.com/en-us/library/cc136787(VS.85).aspx)

## Get-VMNicPort

Returns the switch port object for a Network Interface Card.

### Parameters

|  |  |  |
| --- | --- | --- |
| **NIC** *Required* | (Input from pipeline) | An object representing the a Network Interface Card whose port is sought. |

### Description

Returns the switch port object for a Network Interface Card: This is the logical port on a switch, not the switch itself. It can be used to find the switch.

### Examples

Get-VMNic $core -legacy -VMbus | get-VMNicPort

Returns the switch-ports on the NICs of the VM pointed to by $core.

### See Also:

Select-VMNic

Get-VMNic

Get-VMNICSwitch

## New-VMSwitchPort

Creates a new port on a virtual network switch.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VirtualSwitch**  *Required* | (Input from pipeline) | Either the name of a Network switch, passed as a string, or a WMI object representing an interface. It may be passed via the pipeline. |
| **Name** |  | The identifier for the port, this is normally a GUID and will be created automatically if not passed as a parameter. |
| **Server** |  | Specifies the Hyper-V server to query for a network switch, if the switch is passed as a name. By default "." , the local computer is used. |

### Description

Creates a new port on a virtual network switch.

### Notes

This is intended to be called from other commands and not from the command line.

### See Also:

[CreateSwitchPort Method of the MsVM\_VirtualSwitchManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc136782(VS.85).aspx)

## Get-VMHost

Lists Hyper-V servers registered with Active Directory.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Domain** | (Input from pipeline) | Specifies the Active directory container to search from: defaults to the root domain. |

### Description

Queries Active Directory for Hyper-v connection points - in other words, servers registered with the domain running Hyper-V.

### Examples

get-VMhost "DC=ite,DC=contoso,DC=com" | foreach-object {$\_; Get-VMsummary -server $\_}

Queries the domain ITE.Contoso.com for Hyper-V servers, prints the name of each and dumps the state of its VMs.

## Set-VMHost

Configures the settings for the Hyper-V service itself.

### Parameters

|  |  |  |
| --- | --- | --- |
| **ExtDataPath** |  | If passed, changes the external data path used for Snapshots etc. |
| **VHDPath** |  | If passed, changes the default location for Virtual Hard disk files. |
| **MinMAC** |  | Must be exactly 12 Hexadecimal digits. If set, changes the minimum MAC address for dynamically assigned addresses. |
| **MaxMAC** |  | Must be exactly 12 Hexadecimal digits. If set, changes the maximum MAC address for dynamically assigned addresses. |
| **OwnerContact** |  | Contact information for a person responsible for the server. |
| **OwnerName** |  | Name of a person responsible for the server. |
| **Server** |  | Specifies a one or more Hyper-V server(s) to be updated. By default "." , the local computer, is used.. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Configures the settings for the Hyper-V service, including contact details, MAC Address range, default VHD location, and default data files location.

### See Also:

Get-VMHost

[ModifyServiceSettings Method of the MsVM\_VirtualSystemManagementService Class on MSDN](http://msdn.microsoft.com/en-us/library/cc160709(VS.85).aspx)

## Get-VMLiveMigrationNetwork

Returns a list of Cluster networks in use for live migrating VMs.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Server** |  | Specifies the Hyper-V server / Cluster node to query. By default "." , the local computer is used. |

### Description

Returns a list of Cluster networks in use for live migrating VMs.

### Notes

This command depends on the Cluster PowerShell module being loaded.

### See Also:

Select-VMLiveMigrationNetwork

Get-ClusterResourceType

Get-ClusterParameter

## Select-VMLiveMigrationNetwork

Allows the user to select cluster networks to be used live migrating.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Server** |  | Specifies the Hyper-V server / Cluster node to query. By default "." , the local computer is used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc, with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

Shows a list of clusternetworks from which the user can choose the network(s) to be used for live migration.

### Notes

This command depends on the Cluster PowerShell module being loaded.

### See Also:

Get-VMLiveMigrationNetwork

Get-ClusterNetwork

Get-ClusterResourceType

set-ClusterParameter

## Move-VM

Moves one or more VMs between cluster nodes using Live Migration.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** | (Input from pipeline) Wildcards | The Virtual Machine(s) to migrate. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified all VMs are migrated. |
| **Destination** |  | The name of the cluster node to which Virtual Machines should be migrated. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE. It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. This command has a "ConfirmImpact" level of High: so the default is to request confirmation (unless the $confirmPreference variable is set to "none".). |

### Description

Moves VMs between cluster nodes. If multiple VMs are specified they are moved sequentially. If the cluster commands are not loaded or a VM is not running, or not configured as a cluster application then a warning is returned.

### Examples

Move-VM

Moves all Virtual Machines on the local server to the first other node the cluster - this is only recommended for 2 node clusters.

Get-VM -server NodeA -running | move-VM -Destination NodeB

Gets VMs running on the Server named "Node A" and moves them to the server "nodeB".

### See Also:

Get-VMClusterGroup

Get-VMLiveMigrationNetwork

Set-VMLiveMigrationNetwork

## Get-VMClusterGroup

Returns the Cluster Group of which the VM is a member.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** | (Input from pipeline) Wildcards | The Virtual Machine(s) to test This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is specified all VMs are checked. |
| **Server** |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards cannot be used. |

### Description

Returns the cluster group of which the VM is a member. If the VM is not clustered or if the cluster commands are not loaded the command returns $null. This can be used as a test for Highly-Available VMs.

### Examples

Get-VMClusterGroup "Tenby"

Returns the cluster group for the Virtual Machine named "Tenby" or null if it is not clustered..

Get-VMClusterGroup | where-object {$\_.state -eq "Offline"} | Move-ClusterGroup

Moves offline VMs from to the other node of a cluster. To move running VMs use Move-VM..

### See Also:

Move-VM

Get-VMLiveMigrationNetwork

Set-VMLiveMigrationNetwork

## Sync-VMClusterConfig

Synchronizes the configuration of a VM across all members of a cluster.

### Parameters

|  |  |  |
| --- | --- | --- |
| **VM** | (Input from pipeline) Wildcards | The Virtual Machine(s) to synchronise. This may be a WMI object representing a VM, a string object containing the name of a VM (or a wildcard which matches multiple VMs), or an array containing a mixture of such objects. It may be passed via the pipe. If no VM is passed all VMs will be synchronised. |
| **Server** *Required* |  | Specifies the Hyper-V server on which a named Virtual Machine resides. By default "." , the local computer, is used. An array of server names may be passed but wildcards can not be used. |
| **PSC** |  | THIS PARAMETER SHOULD NEVER BE PASSED FROM THE COMMAND LINE.It is used to ensure correct handling of ‑Whatif, ‑Confirm, ‑Verbose etc., with nested and recursive calls to the command. |
| **Force** |  | Ensures that the user is not prompted before the action is carried out. |

### Description

If the configuration of a clustered VM is changed on the node where it is resident, it is necessary to tell the cluster to update its information so that all nodes are consistent. If this is not done VMs may not be able to migrate. If the failoverClusters commands have not been imported this command generates a warning..

### See Also:

Get-VMClusterGroup

Get-ClusterResource

Update-ClusterVirtualMachineConfiguration

## Select-ClusterSharedVolume

Allows the user to select a clustered shared volume from a list.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Server** |  | Specifies the Hyper-V server on which the cluster shared volume resides. |

### Description

Allows the user to select a clustered shared volume from a list. If cluster functions are not loaded, returns an error.

### Examples

$path = (Select-ClusterSharedVolume).volname ; new-VM -name "Ha" -path $path

Gets the path a of a user determined Cluster shared volume and uses it as the data directory for a new Virtual Machine.

### See Also:

Get-ClusterSharedVolume

## Test-WMIJob

Checks the status of background WMI jobs.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Job** *Required* | (Input from pipeline) | A WMI object representing a job. If a string is passed containing a WMI path it is converted to the WMI object automatically. |
| **StatusOnly** |  | If not specified the Job WMI object is returned. If it is specified only the JobState Property is returned, saving the caller the step of getting the property. |
| **Wait** |  | If not specified the job will only be examined once. If specified it will be examined every 250 milliseconds as long as its state is "Running". |
| **Description** |  | The Description to be displayed while waiting for the job to complete. |

### Description

Some WMI functions create a background job, it may be necessary to wait for a job to complete, or simply to check its current state.

## Test-WMIResult

Checks the result returned by calling a WMI method.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Result** *Required* |  | The WMI Object returned by calling a method of another WMI object. |
| **JobWaitText** |  | The text to be displayed on a progress indicator while waiting for the job to complete (if -wait is specified and a background job exists). |
| **SuccessText** |  | The text written to Verbose: if the method was successful. |
| **FailText** |  | The Text to be included with the error and written to error if the method fails. |
| **wait** |  | If specified will wait for a background job to complete. If not specified a message will be written to Warning indicating the job is still running if it has not completed. |

### Description

When calling a WMI method the result will often by an object containing a return value, which indicates immediate success, the creation of a task to perform the job, or an error. Test-WMI result looks at this code and if necessary uses Test-WMIJob to see what the outcome of the job was.

### Notes

This command is expected to be used by other commands, not from the command line.

### See Also:

Test-WmiJob

## Select-List

Returns an object selected by the user from a table of numbered rows.

### Parameters

|  |  |  |
| --- | --- | --- |
| **InputObject** *Required* | (Input from pipeline) | An array of objects which provide the data for the selection list. The data returned comes from this parameter as well. If a single item is provided, it is returned without showing the list. . |
| **Property** *Required* |  | One or more property names used to format the selection list. . |
| **Multiple** |  | Specifies that multiple items can be selected. |

### Description

Takes a collection of objects and the list of properties to be displayed. This information is used to show a table with numbered rows. The user is prompted to make either a single or multiple choice, by row numbers and the selected objects are returned.

## Select-Item

Returns a zero-based integer indicating the user's selection from a list.

### Description

A list of choices is displayed one after the other, and the users selection is returned as a zero-based integer. Can display a caption followed by a message before the choices.

### Examples

select-item -Caption "Configuring RemoteDesktop" -Message "Do you want to: " -choice " &Disable Remote Desktop","&Enable Remote Desktop","&Cancel" -default 1

will display the following.

Configuring RemoteDesktop Do you want to: [D] Disable Remote Desktop [E] Enable Remote Desktop [C] Cancel [?] Help (default is "E"):.

The function accepts D, E ,or C as input and returns 0 for Disable, 1 for Enable and 2 for Cancel.

Get-choice @{"&Tea"="A drink made from leaves";"&Coffee"="A drink made from beans"}

will display the following.

Please make a selection Choices are presented below [T] Tea [C] Coffee [?] Help (default is "T"):.

Pressing ? will produce help as follows.

T - A drink made from leaves C - A drink made from beans.

### See Also:

[PromptForChoice Method of PSHostUserInterface on MSDN](http://msdn.microsoft.com/en-us/library/system.management.automation.host.pshostuserinterface.promptforchoice(VS.85).aspx)

## Select-Enumtype

Returns a value selected by the user from an Enumeration type.

### Parameters

|  |  |  |
| --- | --- | --- |
| **eType** *Required* |  | The Enumeration type to use to build the selection list. |
| **Default** |  | The Value returned if nothing is selected from the list (if not specified the function will return null). |

### Description

When an enum type has been created to allow easy mapping of names to values, this function prompts the user to select one of the names, and returns the corresponding value.

### Examples

$inboundAction = (Select-EnumType fwaction)

The user is offered choices "Allow" = 1 or "Block" = 0 , the two possible options in the fwAction (firewall action) enum, and the variable is set to 1 or 0, the associated values based on that choice.

## Select-Tree

Allows the user to select from objects in a tree format.

### Parameters

|  |  |  |
| --- | --- | --- |
| **InputObject** *Required* | (Input from pipeline) | The collection of items to output in the tree. May be passed via the pipe For backwards compatibility has an alias of "Items". |
| **StartAt** *Required* |  | The first item in the branch of the tree to output. |
| **Path** |  | The name of the property name which identifies a location in the tree. It does not need to be a fully qualified path, but must be unique. |
| **Parent** |  | The name of the property which holds the path of a node's parent. |
| **Label** |  | The name of the property which holds the text to display. |
| **Indent** |  | The function is called recursively and this parameter specifies how far each branch of the tree is indented. Should not be passed from the command line. |
| **Multiple** |  | Determines whether a single or multiple selection is to made from the tree. |

### Description

The function takes a collection of objects which must have: a property to display (label), a property identify the position in the tree (path) and a property which identifies a node's parent (parent). displays them tree format and prompts the user to choose one (or more than one if -Multiple is specified).

### See Also:

Out-Tree

## Out-Tree

Outputs objects in a tree format.

### Parameters

|  |  |  |
| --- | --- | --- |
| **InputObject** *Required* |  | The collection of items to output in the tree. May be passed via the pipe For backwards compatibility has an alias of "Items". |
| **StartAt** *Required* |  | The first item in the branch of the tree to output. |
| **Path** |  | The name of the property name which identifies a location in the tree. It does not need to be a fully qualified path, but must be unique. |
| **Parent** |  | The name of the property which holds the path of a node's parent. |
| **Label** |  | The name of the property which holds the text to display. |
| **Indent** |  | The function is called recursively and this parameter specifies how far each branch of the tree is indented. It should not be passed from the command line. |

### Description

The function takes a collection of objects which must have: a property to display (label), a property identify the position in the tree (path) and a property which identifies a node's parent (parent).

### Examples

dir -Recurse | Where-Object {$\_.mode -like "D\*"} | out-tree -start (get-item $pwd) -path pspath -parent psparentPath -label name

Outputs a tree of subfolders of the current folder; the starting item is the current folder, for each item in thee tree its path is stored in the PsPath property, and its parent is stored in the PsParentPath propery. The name property is output in the tree view.

### See Also:

Select-Tree

## Test-Admin

Checks to see if the current session has administrator privileges.

### Description

Some tasks require administrator privileges, and on Windows Vista/Server 2008 and later this may require the user to choose "Run as administrator" when starting PowerShell The command returns a Boolean indicating the elevated status - true = elevated, false = not elevated.

### Examples

Test-Admin

Returns true if the session was started as administrator, false otherwise.

## Convert-DiskIDtoDrive

Converts a logical disk index to a drive letter.

### Description

Converts a logical disk index to a drive letter.

### Examples

PS > Convert-DiskIDToDrive 0

Returns the drive letters of the partitions on the specified disk. For Example C:, D:.

## Get-FirstAvailableDriveLetter

Returns a CHAR indicating the first available drive letter.

### Description

Returns a CHAR indicating the first available drive letter.

## A quick guide to the PowerShell code in the module

PowerShell V2 has 236 cmdlets, and 138 aliases and 41 functions present before any Modules are loaded. Only 36 of these are used in the Module, and of those the most popular 18 account for 90% of all the calls made.

|  |  |
| --- | --- |
| Working with objects | |
| Get-WmiObject | Gets instances of Windows Management Instrumentation (WMI) classes or information about the available classes. |
| New-Object | Creates an instance of a Microsoft .NET Framework or COM object. |
| ForEach-Object | Performs an operation against each of a set of input objects. |
| Where-Object | Creates a filter that controls which objects will be passed along a command pipeline. |
| Select-Object | Selects specified properties of an object or set of objects.  It can also select unique objects from an array of objects, or it can select a specified number of objects from the beginning or end of an array of objects. |
| Measure-Object | Calculates the numeric properties (count, sum, average) of objects, and the characters, words, and lines in string objects, such as files of text. |
| Sort-Object | Sorts objects by property values. |
| Add-Member | Adds a user-defined custom member to an instance of a Windows PowerShell object. |
| Working with the file-system / registry | |
| New-PSDrive | Creates a Windows PowerShell drive in the current session. |
| Join-path | Combines a path and a child path into a single path. |
| Resolve-Path | Resolves the wildcard characters in a path, and returns the path contents. |
| Test-path | Determines whether all elements of a path exist. |
| Split-Path | Returns the specified part of a path. |
| Get-ChildItem | Gets the items and child items in one or more specified locations. |
| Get-ItemProperty | Gets the properties of a specified item. |
| Set-Content | Writes or replaces the content in an item with new content. |
| Working with date and time | |
| Start-sleep | Suspends the activity in a script or session for the specified period of time. |
| Get-Date | Gets the current date and time. |
| New-TimeSpan | Creates a TimeSpan object. |
| Handling Input and out put | |
| Read-Host | Reads a line of input from the console. |
| Write-Host | Writes customized output to the console. |
| Write-Warning | Writes a message to the warning Message stream. |
| Write-Debug | Writes a message to the debug message stream. |
| Write-verbose | Writes text to the verbose message stream. |
| Write-Error | Writes an object to the error stream. |
| Write-Progress | Displays a progress bar within a Windows PowerShell command window. |
| Out-Null | Deletes output instead of sending it to the console. |
| Out-Default | Sends the output to the default formatter and to the default output stream. |
| Out-Host | Sends output to the console. |
| Format-Table | Formats the output as a table. |
| Working with commands and datatypes | |
| Get-Module | Gets the modules that have been imported or that can be imported into the current session. |
| Get-Command | Gets basic information about cmdlets and other elements of Windows PowerShell commands. |
| Set-Alias | Creates or changes an alias (alternate name) for a command in the current Windows PowerShell session. |
| Invoke-Expression | Runs PowerShell commands or expressions |
| Start-Process | Starts one or more processes on the local computer. |
| Add-Type | Adds a Microsoft .NET Framework type (a class) to a Windows PowerShell session. |

Almost all the work of the module revolves around working with WMI objects, by way of illustration, here is a simplified version of the Get-VM function.

Function Get-VM

{ param(

[parameter(ValueFromPipeLine = $true)] [ValidateNotNullOrEmpty()][Alias("VMName")]

$Name = "%",

$Server = "."

)

Process {

if ($Name.count -gt 1 ) {

[Void]$PSBoundParameters.Remove("Name")   
 $Name | ForEach-object {Get-VM -Name $\_ @PSBoundParameters}

}

if ($name -is [String]) {

$Name = $Name.Replace("\*","%")

$WQL = "SELECT \* FROM MSVM\_ComputerSystem WHERE ElementName LIKE '$Name' AND ProcessID >= 0"

Get-WmiObject -computername $Server -NameSpace $HyperVNamespace -Query $WQL |   
 Add-Member -MemberType ALIASPROPERTY -Name "VMElementName" -Value "ElementName" -PassThru

}

}

}

First the function declares a parameter $Name with a default value of "%", $Name might be a string, an array of strings or a WMI object (which the full version of function checks for and returns the associated VM). Therefore the $Name parameter isn’t cast as any type. It allows the name to be passed via the pipeline, and allows it to be called VMName (or a shortening such as VM). A second Parameter $Server specifies where the command is to run.

A function can have 3 blocks, named Begin, Process and End. Begin is run once when the function is invoked, Process is re-run for each successive object passed via the pipeline and End is run once on completion. If none of these blocks are specified the function body is treated as an End block. To make sure all Get-VM will handle multiple objects passed via from the pipeline, all the work of the function is done as a process block.

If $name is an array with multiple elements, the function is called recursively for each one. To simplify the call many functions use the same pattern: Strip the parameter which is changing ($name in this case) from the array $PSBoundParameters, and then call the function once for each value using what remains of $PSBoundParameters.

If $name is a single string the \* wildcard is replaced with the % wildcard used in WMI’s query language *WQL*. A query is built up – in the full version of Get-VM additional parameters specify if only *Running*, *Stopped*, or *Suspended* VMs are to be fetched, so the query can be more complex than seen here.

Get-WMIObject is called specifying the computer which should be queried. This can be a single string ("." the local machine by default) or an array. The WMI namespace and WQL query are provided and if matching VMs are found Get-WMIObject returns objects representing them. These objects are then piped into Add-Member; this cmdlet adds an *alias property* to the WMI object, it is named "*VMElementName*" and maps through to the *ElementName* property. This is done because some VM components have an element name such as "Network Card", and it Is useful to have the element name for their VM, so VMElementname is added at numerous places in the library, and to allow consistency it is provided for VMs themselves.

Let’s look at a two more, closely related, functions. Stop-VM and Set-VM state

Function Stop-VM

{ [CmdletBinding(SupportsShouldProcess=$True , ConfirmImpact='High')]

Param(

[parameter(Mandatory = $true, ValueFromPipeline = $true)]

$VM,

[parameter()][ValidateNotNullOrEmpty()]

$Server = "." ,

[Switch]$Wait,

$PSC,

[Switch]$force

)

Process {

if ($psc -eq $null) {$psc = $pscmdlet}  
 if (-not $PSBoundParameters.psc) {$PSBoundParameters.add("psc",$psc)}

Set-VmState -State ([VMState]::Stopped) @PSBoundParameters }

}

}

As you can see Stop-VM starts by saying it supports *Should Process* which allows it to take the ‑WhatIf, ‑Confirm, and ‑Verbose switches. In addition by saying its impact for confirmation purposes is *high*, unless the user changes the value of the $ConfirmPreference variable it will prompt before continuing.

It takes a -VM parameter which must be present and might come from the pipeline, a server parameter as before, and a wait switch – without which the command will return while the request to stop the VM is still being processed. Finally it takes a linked pair of parameters -PSC and -Force, which are explained in a moment.

As with Get-VM, the work is done in a Process block which allows multiple VMs to be piped into Stop-vm. The parameters handed to Stop-VM are passed on to Set-VMState with a constant – defined in a *enumeration* type created by the module – which says the state we want the VM set to is "Stopped".

The "Should process" behaviour is handled by an automatic variable $Pscmdlet which has a shouldProcess() method; this returns true to say execution should continue to make a change and false to say it should not. Calling shouldProcess() may ask the user to confirm something (with the ‑confirm switch), output a verbose message saying what was being done (with the ‑verbose switch), or output a message saying what would have been done, without actually doing it (with the ‑whatif switch). However $PsCmdlet is created for each call to a function, so $PSC is passed between functions. That way a single instance is used which results in the right behaviour if the user selects "No to all" or "Yes to all", and allowing a "Confirm impact" level to be specified in one function and honoured by another. If $PSC isn’t passed as a parameter it is set to $PsCmdlet, and added to the list of $PSBoundParameters. The -force switch is used in conjunction with $PSC if –force is specified the command will be run without prompting the user.

The final piece of is to look at Set-VMState

$lStr\_VMStateChanging = "Change state to {0}"

$lStr\_VMStateChangeFail = "Failed to change state of VM {0} to: {1}"

$lStr\_VMStateChangeSuccess = "Sucessfully changed state of VM {0} to: {1}"

$lStr\_VMStateWaiting = "Change state of VM {1} to {0} "

Function Set-VMState

{ CmdletBinding(SupportsShouldProcess=$true)]

param(

[parameter(Mandatory = $true, ValueFromPipeline = $true)]

$VM,

[parameter(Mandatory = $true)]

[VMState]$State,

[parameter()][ValidateNotNullOrEmpty()]

$Server = "."

[switch]$Wait,

$PSC,

[switch]$Force

)

Process {

if ( $psc -eq $null) { $psc = $pscmdlet  
 $PSBoundParameters.add("psc",$psc) }

if ( $VM -is [String]) { $VM = $(Get-VM -Name $VM -Server $Server)}

if ( $VM.count -gt 1 ) {

[Void]$PSBoundParameters.Remove("VM")   
 $VM | ForEach-object {Set-VmState -VM $\_ @PSBoundParameters}

}

if (($vm.\_\_CLASS -eq 'Msvm\_ComputerSystem') -and ($force –or  
 $psc.shouldProcess($vm.ElementName, ($lStr\_VMStateChanging -f $State)))

) {

$VM.RequestStateChange($State) |  
 Test-wmiResult -wait:$wait `   
 -JobWaitText ($lStr\_VMStateWaiting -f $State, $Vm.elementName ) `

-SuccessText ($lStr\_VMStateChangeSuccess -f $Vm.elementName,$State)`

-failText ($lStr\_VMStateChangeFail -f $Vm.elementName,$state)

}

}

}

The parameters for Set-VMState are almost identical to those from Stop-VM except that there is an extra mandatory parameter -State. This uses the [VMState] *enum* type so can be passed as 3 (the numeric value meaning stopped) or the text "Stopped". You can see that as before everything happens in a Process block, and if PSC is not passed it is set to $PsCmdlet and added to the bound parameters.

If the $VM holds a string it gets resolved to zero or more VMs by calling Get-VM.

If $VM is an array (either because it was passed as one, or because Get-VM returned one), then the function calls itself recursively – stripping the -VM parameter off the list of $PsBoundParameters and replacing it with each value in turn.

If the $VM is a WMI object with a \_\_CLASS property of "Msvm\_ComputerSystem" then the next step is to check to see if the -FORCE switch was passed, and if not, to evaluate $Psc.ShouldProcess(). This method takes two parameters: the name of what is being affected, and what is being doing to it. The module uses a string (in practice the strings are gathered into a central file) and the –f operator substitutes the state into the placeholder {0} in the text "Change state to {0}".

If -force or shouldprocess() gives clearance to make the change the VM object’s RequestStateChange() method is called with the requested state. PowerShell will convert the Enum to the integer required by the method.

The result returned by RequestStateChange() is not particularly user friendly – it is basically a code indicating "success", "running in the background", or "failed." Test-WMIresult is another function in the module which takes the result , and a –wait switch (which was passed to Stop-VM and on to Set‑VMState). If the task is still running and -wait is specified the function loops until the result changes to completed; once it has completed the function returns more meaningful text for the result.

It is worth taking a moment to emphasise something in Set-VMstate – the module goes to considerable lengths to be flexible in its use of parameters, for example if you are running a cluster you might not know if the VM named "*Paris*" is on Node1 or Node2, so anywhere that you need to specify a VM you can use   
–vm "Paris" –server "node1","node2"

Except in a few cases (like attaching a Virtual Hard disk), if it make sense to carry out an action on multiple VMs you can write *"London","Paris"* If you have a list of VM names in a file it seems logical that you can do  
Get-Content List.txt | Start-VM

and the module allows this. Finally where a command returns objects representing VMs or their components then it makes it be possible pipe those objects into commands which use them.

One last thing to look at is how *Resource Allocation Setting Data* (RASD) objects are used to configure elements of a VM, such as memory

Function Set-VMMemory

{[CmdletBinding(SupportsShouldProcess=$true)]

param(

[parameter(Mandatory = $true, ValueFromPipeLine = $true)]

$VM,

[parameter(Mandatory = $true)][Alias("MemoryInBytes")]

[long]$Memory,

$Server = ".",

$PSC,

[switch]$Force

)

process {

if ($psc -eq $null) {$psc = $pscmdlet} ;

if (-not $PSBoundParameters.psc) {$PSBoundParameters.add("psc",$psc)}

if ($VM -is [String]) { $VM = Get-VM -Name $VM -Server $Server}

if ($VM.count -gt 1 ) {

[Void]$PSBoundParameters.Remove("VM")   
 $VM | ForEach-object {Set-VMMemory -VM $\_ @PSBoundParameters}

}

if ($vm.\_\_CLASS -eq 'Msvm\_ComputerSystem') {

if (-not ($memory % 2mb)) {$memory /= 1mb}

$SettingData = Get-VMMemory -VM $VM

$SettingData.Limit = $SettingData.Reservation = $SettingData.VirtualQuantity = $Memory

Set-VMRASD -vm $vm -rasd $SettingData -psc $psc

}

}

}

The parameters and support for should process are as before. The only parameter which is new is -memory, and as before the $PSC / $PSCCmdlet are handled for *SupportsShouldProces*s, Strings representing the VM(s) are converted and the function is called recursively to support multiple VMs, it only does any real processing if it has an *Msvm\_ComputerSystem* WMI object,

The function allows the user to write the –memory parameter as 512MB or 0.5 GB (PowerShell will convert those to 536870912), but the WMI objects work in MB so the function takes anything which is divisible by 2MB , and divides $memory down to get the intended number.

It then calls Get-VMMemory to get a setting data object which represents memory on the VM. It sets 3 properties of this object to the value in $memory. Limit, reservation and quantity are there to support dynamic memory although this feature is not implemented at the time of writing.

Then the function calls Set-VMRasd , passing it the setting data (which in this case is memory, but could be any virtual hardware) the VM, and $PSC so that shouldProcess() behaves correctly.

Set-VMRasd (like its companions Add-VMRasd and Remove-VMRasd) uses a method of the MSVM\_VirtualSystemManagementService object to change a VM. It wraps to the call to this method in the same should process construction we saw with Set-VMState, and sends the result to Test-WMIresult as Set-VMstate did.

Because so many of the functions depend on getting the right kind of WMI object it is worth just taking a moment to explain the behaviour and use of Get-WMIObject.

WMI objects form a hierarchy, a computer will have multiple *namespaces* each of which contains multiple *classes.* There is no guarantee that there will be any instances of a given class.

You can get all the WMI objects of a given class with the syntax

Get-WMIObject [-computerName server] [-namespace Name] [-Class] classname ­[-filter condition]

For example: Get-WmiObject win32\_logicaldisk

In this case the local computer and default name space are assumed, and power assumes the only parameter is the class name.

$d = Get-WmiObject win32\_logicaldisk -Filter "DeviceID='C:'"

Reduces the results to a single disk, and stores them in a variable. This can also be written in a SQL-style syntax, known as WQL. This would appear as follows.

$d = Get-WmiObject -query "Select \* from win32\_logicaldisk where DeviceID='C:' "

The two versions can be used interchangeably depending on which is the most convenient at the time – the module generally uses the second one.

In Hyper-V it is often necessary to find objects based on their relationship, using an associators query for example

Get-WmiObject -Query "Associators of {$($d.\_\_Path)}"

In fact the way powershell uses the \_\_path property of a WMI object when expanding it in a string so it can be written more simply as

Get-WmiObject -Query "Associators of {$d}"

You can check what classes are associated with an object by using the syntax

Get-WmiObject -Query "Associators of {$d} where classdefsonly"

And get the objects of a specific class like this

Get-WmiObject -Query "Associators of {$d} where resultclass=Win32\_computerSystem"

Notice that *where* behaves differently in an associators query, it can’t be used for filtering in the same way that it can in a select.

PowerShell V2 introduced a shorthand for this which is $d.GetRelated("win32\_computerSystem")

This is used in several places in the module, but to friendly to PowerShell remoting (which serializes objects to pass them over the network, so they cease to be "true" WMI objects) the first form of the syntax is used whenever $d is passed as a parameter and the getRelated() form only when it is obtained inside the function.

# WMI Object classes used by the functions

|  |  |
| --- | --- |
| **WMI Class name** | **Used in Functions** |
| Cim\_Datafile | Get‑VHD, Show‑VHDMenu, Show‑VHDMenu, Show‑VHDMenu, Show‑VHDMenu, Show‑VHDMenu |
| Msvm\_ActiveConnection | Show‑HypervMenu |
| Msvm\_AllocationCapabilities | New‑VMRASD |
| Msvm\_BindsTo | Get‑VMNICVLAN, Set‑VMNICVLAN, Set‑VMNICVLAN |
| Msvm\_ComputerSystem | Add‑VMNewHardDisk, Add‑VMDisk, Add‑VMDisk, Add‑VMDrive, Add‑VMFloppyDisk, Add‑VMPassThrough, Add‑VMSCSIController, Get‑VMDisk, Get‑VMFloppyDisk, Remove‑VMDrive, Remove‑VMFloppyDisk, Set‑VMDisk, Show‑VMMenu, Add‑VMNIC, Get‑VMNIC, Set‑VMNICAddress, Set‑VMNICSwitch, Get‑VMSnapshot, Get‑VMSnapshotTree, New‑VMSnapshot, Rename‑VMSnapshot, Restore‑VMSnapshot, Select‑VMSnapshot, Update‑VMSnapshot, Export‑VM, Get‑VM, Get‑VM, Get‑VM, Get‑VMBuildScript, Get‑VMClusterGroup, Get‑VMSummary, Get‑VMThumbnail, Invoke‑VMShutdown, Move‑VM, New‑VMConnectSession, Ping‑VM, Remove‑VM, Set‑VM, Set‑VM, Set‑VMState, Test‑VMHeartBeat, Get‑VMIntegrationComponent, Get‑VMKVP, Get‑VMProcessor, Get‑VMSettingData, Remove‑VMKVP, Set‑VMCPUCount, Set‑VMMemory, Set‑VMSerialPort, Sync‑VMClusterConfig |
| Msvm\_DiskDrive | Select‑VMPhysicalDisk, Show‑VMDiskMenu, Show‑VMDiskMenu |
| Msvm\_EmulatedEthernetPortSettingData | Get‑VMByMACAddress, Get‑VMNIC |
| Msvm\_ExternalEthernetPort | Show‑HypervMenu, Show‑HypervMenu, New‑VMExternalSwitch, New‑VMExternalSwitch, Select‑VMExternalEthernet, Set‑VMNICVLAN |
| Msvm\_HeartbeatComponent | Test‑VMHeartBeat |
| Msvm\_HostedAccessPoint | Set‑VMNICVLAN |
| Msvm\_InternalEthernetPort | Show‑HypervMenu, Remove‑VMSwitchNIC |
| Msvm\_KvpExchangeComponent | Get‑VMKVP |
| Msvm\_KvpExchangeDataItem | Add‑VMKVP, Remove‑VMKVP |
| Msvm\_MemorySettingData | Get‑VMMemory |
| Msvm\_MountedStorageImage | Get‑VHDMountPoint, Mount‑VHD |
| Msvm\_NetWorkElementSettingData | Get‑VMNICVLAN, Set‑VMNICVLAN, Set‑VMNICVLAN |
| Msvm\_PreviousSettingData | Get‑VMSnapshot |
| Msvm\_Processor | Get‑VMProcessor |
| Msvm\_ProcessorSettingData | Get‑VMCPUCount |
| Msvm\_ResourceAllocationSettingData | Add‑VMFloppyDisk, Get‑VMDiskByDrive, Get‑VMDiskByDrive, Get‑VMDiskController, Get‑VMDriveByController, Get‑VMDriveByController, Get‑VMFloppyDisk, Remove‑VMDrive, Set‑VMDisk, Show‑VMDiskMenu, Show‑VMDiskMenu, Show‑VMDiskMenu, Get‑VMSerialPort |
| Msvm\_ResourcePool | Select‑VMPhysicalDisk, Show‑VMDiskMenu, Show‑VMDiskMenu |
| Msvm\_SettingsDefineCapabilities | New‑VMRASD |
| Msvm\_ShutdownComponent | Invoke‑VMShutdown |
| Msvm\_SwitchLANEndpoint | New‑VMInternalSwitch, Set‑VMNICVLAN |
| Msvm\_SwitchPort | Show‑HypervMenu, Set‑VMNICVLAN |
| Msvm\_SyntheticEthernetPortSettingData | Get‑VMByMACAddress, Get‑VMNIC |
| Msvm\_VirtualSwitch | Get‑VMNICSwitch, Get‑VMSwitch, New‑VMSwitchPort, Remove‑VMSwitch |
| Msvm\_VirtualSwitchManagementService | New‑VMExternalSwitch, New‑VMInternalSwitch, New‑VMPrivateSwitch, New‑VMSwitchPort, Remove‑VMSwitch, Remove‑VMSwitchNIC, Set‑VMNICSwitch |
| Msvm\_VirtualSystemGlobalSettingData | Show‑VMMenu, Get‑VMBuildScript, New‑VM, Set‑VM |
| Msvm\_VirtualSystemManagementService | New‑VMSnapshot, Remove‑VMSnapshot, Restore‑VMSnapshot, Export‑VM, Get‑VMSummary, Import‑VM, New‑VM, Remove‑VM, Set‑VM, Set‑VMHost, Add‑VMKVP, Add‑VMRASD, Remove‑VMKVP, Remove‑VMRASD, Set‑VMRASD |
| Msvm\_VirtualSystemManagementServiceSettingData | Get‑VhdDefaultPath, Show‑HypervMenu, Show‑HypervMenu, Set‑VMHost |
| Msvm\_VirtualSystemSettingData | Get‑VMByMACAddress, Set‑VMNICAddress, Set‑VMNICSwitch, Get‑VMSnapshot, New‑VMSnapshot, Remove‑VMSnapshot, Rename‑VMSnapshot, Restore‑VMSnapshot, Get‑VM, Get‑VM, Get‑VMSettingData |
| Msvm\_VlanEndpoint | Remove‑VMSwitch |
| Win32\_CdromDrive | Add‑VMDisk, Add‑VMDisk, Set‑VMDisk, Show‑VMDiskMenu |
| Win32\_Directory | Get‑VHD, Show‑VHDMenu |
| Win32\_DiskDrive | Get‑VHDMountPoint, Mount‑VHD |
| Win32\_DiskPartition | Get‑VHDMountPoint, Convert‑DiskIDtoDrive, Convert‑DiskIDtoDrive |
| Win32\_LogicalDisk | Wait‑ForDisk, Convert‑DiskIDtoDrive, Get‑FirstAvailableDriveLetter |
| Win32\_OperatingSystem | Show‑VMDiskMenu |
| Win32\_PingStatus | Ping‑VM |